НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Автоматизації проектування енергетичних процесів і систем

«На правах рукопису»
УДК _____

«До захисту допущено»
Завідувач кафедри
_____ /затв. О.В. Коваль /
“____” _____ 2018 р.

Магістерська дисертація
на здобуття ступеня магістра

зі спеціальності ***121 – Інженерія програмного забезпечення***
спеціалізації ***Програмне забезпечення розподілених систем та Web-технологій***

на тему: Застосування багатопотоковості для навчання нейронної мережі

Виконав (-ла): студент (-ка) *6* курсу, групи *ТВз-71мп*

Сініцин Володимир Русланович _____
(прізвище ім’я, по батькові) (підпис)

Науковий керівник *к.т.н. Смаковський Д.С.* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний
(повна назва)

Кафедра автоматизації проектування енергетичних процесів і систем
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
«___»_____20___ р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Сініцин Володимиру Руслановичу
(прізвище, ім’я, по батькові)

1. Тема дисертації: Застосування багатопотоковості для навчання нейронної мережі

науковий керівник дисертації Смаковський Денис Сергійович, к.т.н.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від « » 2018 р. № _____

2. Термін подання студентом дисертації: 10 грудня 2018 року

3. Об’єкт дослідження: Алгоритми навчання нейронних мереж

4. Предмет дослідження: Паралельні алгоритми навчання нейронних мереж

5. Перелік питань, які потрібно розробити:

5.1. Літературний огляд нейронних мереж.
5.2. Аналіз алгоритмів навчання нейронних мереж.
5.3 Аналіз алгоритмів паралелізації нейронних мереж.
5.4. Вибір та опис засобів програмної реалізації.
5.5. Опис програмної реалізації.
5.6. Створення програмного продукту.
5.7. Отримання результатів.
5.8. Розробка стартап-проекту.

6. Орієнтовний перелік ілюстративного матеріалу:

6.1. Презентація PowerPoint відповідно до теми дисертації.

7. Орієнтовний перелік публікацій:

7.1. Сініцин В.Р., к.т.н. Смаковський Д.С. Обробка даних за допомогою нейронної мережі прямого розповсюдження.

8. Дата видачі завдання «____»_____201___р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
	Затвердження теми роботи	17.05.2018	
	Вивчення та аналіз задачі. Проведення дослідження по вибраній темі	01.06.2018-03.09.2018	
	Розробка архітектури та загальної структури системи	03.09.2018-28.09.2018	
	Програмна реалізація системи	01.10.2018-26.10.2018	
	Захист програмного продукту	22.10.2018	
	Оформлення пояснювальної записки	02.09.2018-10.12.2018	
	Передзахист	26.11.2018-30.11.2018	
	Захист	17.12.2018	

Студент _____ Сініцин В.Р._____
(підпис) (прізвище та ініціали)

Науковий керівник _____ Смаковський Д.С._____
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Магістерська дисертація складається зі вступу, семи розділів, висновку, переліку посилань з 32 найменувань, 2 додатки, і містить 36 рисунки, 23 таблиці. Повний обсяг магістерської дисертації складає 112 сторінки, з яких перелік посилань займає 3 сторінки, додатки – 12 сторінок.

Об’єкт дослідження – алгоритми навчання нейронних мереж

Предмет дослідження – паралельні алгоритми навчання нейронних мереж.

Метою даної магістерської дисертації є розробка методу багатопотокового навчання нейронної мережі та реалізування його в програмному продукті.

Розроблений алгоритм багатоптокового навчання повинний проводити навчання швидше за нинішні алгоритми паралелізації. Також за цієї умови нейронна мережа повинна піддаватися навчанню. Отримані результати показали перевагу в швидкості навчання обраного методу багатопотоковості в порівнянні із класичним методом. Даний метод змінює алгоритм навчання тому питання використання цього методу потрібно досліджувати в подальшому

НЕЙРОННА МЕРЕЖА, НАВЧАННЯ НЕЙРОННОЇ МЕРЕЖІ, ПАРАЛЕЛЬНІ МЕТОДИ.

ABSTRACT

Master's thesis consists of an introduction, seven chapters, conclusion, list of references with 32 titles, 2 annexes, and contains 36 figures, 23 tables. The full range of master's thesis is 112 pages with a list of links takes 3 pages, apps - 12 pages.

The object of study – algorithms for training neural networks.

The subject of research is the parallel algorithms for training neural networks.

The purpose of this master's thesis is to develop a multi-threaded neural network teaching method and implement it in a software product.

The developed algorithm of multistep training should carry out training faster than current algorithms of parallelization. Also under this condition, the neural network should be trained. The obtained results showed the advantage of learning the chosen method of multithreading in comparison with the classical method. This method changes the learning algorithm so the question of using this method should be investigated in the future

NEURAL NETWORK, NEURAL NETWORK TRAINING, PARALLEL METHODS.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ ТА СКОРОЧЕНЬ

БД	-	База даних.
ШНМ	-	Штучна нейронна мережа.
НМ	-	Нейронна мережа.
РМ	-	Розрахункова модель.
ОЯ	-	Оцінка якості.
ШПФ	-	Швидке преобразування Фурє.
ДН	-	Динамічне навчання.
СН	-	Статичне навчання.
СКБД	-	Система керування базами даних.
ПК	-	Персональний комп’ютер.
ЕОМ	-	Електронна обчислювальна машина.
SQL	-	Structured query language — мова структурованих запитів.
HTML	-	Мова розмітки гіпертекстових документів.
IDE	-	Інтегроване середовище розробки.
XML	-	Розширювана мова розмітки.
UI	-	Інтерфейс користувача.
LIQ	-	Language Integrated Query — мова запитів.
AJAX	-	Асинхронний Javascript і XML.
FA	-	Функція активації

ВСТУП

Нейронна мережа є універсальним інструментом для обробки даних. Ці данні можуть відноситись до будь яких галузей: економіка, біологія, фізика, аграрія, медицина. Основна її перевага це відокремлення залежностей між вхідними даними та одержаним результатом. Нейронні мережі являються потужним методом моделювання, що дозволяє розшифровувати складні залежності.

Основною проблемою нейронної мережі є складність її навчання. Це пов’язано із великою кількостю залежних параметрів. Тому на цей процес витрачається багато часу та ресурсів.

В умовах стрімкого розвитку інформаційних технологій у сфері нейронних мереж та їх ускладнення, актуальним є використання багатопотоковості для пришвидшення процесу навчання нейронної мережі.

Метою даної магістерської дисертації є розробка методу багатопотокового навчання нейронної мережі та реалізування його в програмному продукті.

До задач які повинні бути виконанні для досягнення мети слід віднести аналіз існуючих підходів до проектування та паралелізації нейронних мереж, після чого на основі результатів попереднього аналізу розробити агоритм роботи та структуру програми, реалізувати та протестувати програму для паралельного та послідовного навчання нейронної мережі.

Розроблений алгоритм багатоптокового навчання повинний проводити навчання швидше за нинішні алгоритми паралелізації. Також за цієї умови нейронна мережа повинна піддаватися навчанню.

1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО НЕЙРОННУ МЕРЕЖУ

Нейронні мережі - це моделі біологічних нейронних мереж мозку, в яких нейрони імітуються відносно простими, часто однотипними, елементам. Такі нейрони називаються штучними. Наука нейронних мереж існує досить давно, однак саме в зв'язку з останніми досягненнями науково-технічного прогресу ця сфера починає набувати популярності.

1.1. Історія нейронної мережі

Ідея нейронних мереж народилася в рамках теорії штучного інтелекту, в результаті спроб імітувати здатність біологічних нервових систем навчатися і виправляти помилки.

Початок теорії нейронних мереж і нейрокомп'ютерів поклала робота американських нейрофізіологів Мак-Каллока і Піттса «Логічне числення ідей, що відносяться до нервової діяльності» (1943), в якій вони запропонували математичну модель біологічного нейрона.

Серед основних робіт слід виділити модель Д. Хеба, який в 1949 році запропонував закон навчання, який був стартовою точкою для алгоритмів навчання штучних нейронних мереж.

Нейронні мережі з'явилися в теорії управління одним з перших прикладів переходу від управління найпростішими лінійними стаціонарними системами до управління складними нелінійними, нестаціонарними, багатовимірними, багатозв'язними системами. У другій половині 1960-х років народилася методика синтезу нейронних мереж, яка розвивалася і успішно застосовувалася протягом наступних майже п'ятдесяти років. [1]

Введення зворотних зв'язків і, як наслідок, розробка алгоритмів настройки їх коефіцієнтів в 1960-80 роки мали чисто теоретичний сенс, так як не було практичних завдань, які могли задовольняти такі структури. Лише в кінці 1980-х – початку

1990-х років стали з'являтися такі завдання і найпростіші структури з налаштованим зворотними зв'язками для їх вирішення (так звані рекурентні нейронні мережі).

Розробники в області нейромережевих технологій займалися не тільки створенням алгоритмів настройки багатошарових нейронних мереж і нейромережевими алгоритмами вирішення різних завдань, але і найбільш ефективними (на поточний момент розвитку технології електроніки) апаратними емуляторами (особливі програми, які призначені для запуску однієї системи в оболонці іншого) нейромережевих алгоритмів.

У 1960-і роки, до появи мікропроцесора, найбільш ефективними емуляторами нейронних мереж були аналогові реалізації розімкнутих нейронних мереж з розробленими алгоритмами налаштування на універсальних ЕОМ (іноді системи на адаптивних елементах з аналогової пам'яттю). Такий рівень розвитку електроніки робив актуальним введення перехресних зв'язків в структурі нейронних мереж. Це призводило до значного зменшення числа нейронів в нейронній мережі при збереженні якості виконання завдання (наприклад, дискримінантної здатності при вирішенні задач розпізнавання образів).

На подальший розвиток теорії нейронної мережі істотний вплив зробила монографія американського нейрофізіолога Ф. Розенблатта «Принципи нейродинаміки», в якій він детально описав схему перцептрона (пристрої, що моделює процес сприйняття інформації людським мозком). Його ідеї отримали розвиток в наукових роботах багатьох авторів. У 1985-86 рр. теорія нейронних мереж отримала «технологічний імпульс», викликаний можливістю моделювання нейронних мереж на що потужних персональних комп'ютерах.

Теорія нейронної мережі досить активно розвивається в 21 столітті. За оцінками фахівців, найближчим часом очікується значний технологічний зростання в області проектування нейронних мереж і нейрокомп'ютерів. За останні роки вже відкрито чимало нових можливостей нейронних мереж, а роботи в даній області вносять істотний внесок в промисловість, науку і технології, мають велике економічне значення.

1.2. Предмет цілі та задачі нейронної мережі

Нейронна мережа є універсальним інструментом для обробки даних. Ці данні можуть відноситись до будь яких галузей: економіка, біологія, фізика, аграрія, медицина. Основна її перевага це відокремлення залежностей між вхідними даними та одержаним результатом. Нейронні мережі являються потужним методом моделювання, що дозволяє розшифровувати складні залежності.

Перелік задач, які можна вирішити за допомогою нейронної мережі, досить великий. Він визначається тим, як мережа працює і тим, як вона навчається. Таким чином, нейронну мережу слід застосовувати в ситуації, коли є відома інформація, і потрібно з неї отримати результат. [2]

До числа завдань які може вирішити нейронна мережа можна відносити завдання розпізнавання образів і прогнозування подій, зокрема, знаходження тенденції зміни даних.

У нинішні часи штучний інтелект на основі нейронних мереж широко використовується при аналізі прихованих закономірностей для виявлення в історичних даних таких шаблонів, якими можна керуватися в майбутньому. Наприклад, аналіз прихованих закономірностей в даних може застосовуватися в банківській установі для визначення характеристик потенційного зловмисника серед позичальників.

Крім того, нейронні мережі використовуються, як інтерфейсна частина в експертних системах, що обробляють великі обсяги вхідних даних від датчиків, і від яких вимагається реакція в реальному часі. Нейронна мережа в такому випадку відсіювала дані які вважаються некоректними та можуть в значній мірі повипливати на отриманий результат.

Зокрема, в 1980-х роках нейронна мережа, яка експлуатувалася на звичайному мікрокомп'ютері, дозволила отримати дуже якісне рішення задачі комівояжера за 0,1 секунди. Цей результат набагато перевищував час отримання оптимального рішення

із застосуванням традиційної алгоритмічної системи на аналогічному обладнанні, який становив, в те, час одна година.

Слід зазначити, що завдання комівояжера має важливе практичне значення, оскільки є класичною задачею, яку доводиться вирішувати при маршрутизації пакетів в системі передачі даних. Завдання пошуку оптимальних маршрутів є важливим засобом мінімізації часу, оскільки від цього залежать ефективність і швидкодію, як при маршрутизації пакетів даних через Інтернет, так і при доставці посилок поштою численним адресатам.

Нейронна мережа являє собою сукупність нейронів, які становлять шари. У кожному шарі нейрони між собою ніяк не пов'язані, але пов'язані з нейронами попереднього і наступного шарів. Інформація надходить з першого на другий шар, з другого - на третій і т.д.

Кількість шарів і нейронів у них визначають точність і достовірність отриманих результатів при вирішенні завдань. Таким чином чим більше шарів і нейронів на кожному шарі - тим менше помилок і вище надійність роботи мережі. Однак, якщо побудувати занадто велику мережу, то можна зіткнутися зі зменшенням продуктивності і збільшенням складності моделі. Тому при виборі архітектури мережі слід брати до уваги умови розв'язуваної задачі.

Штучні нейронні мережі імітують поведінку мозку в простому вигляді. Вони можуть бути навчені контрольованим і неконтрольованим шляхами. У контрольованій нейронній мережі, навчання відбувається шляхом передачі відповідної вхідної інформації і прикладів вихідної інформації. Наприклад, спам-фільтр в електронній поштовій скриньці: вхідний інформацією може бути список слів, які зазвичай містяться в спам-повідомленнях, а вихідною інформацією - класифікація для повідомлення (спам, не спам). Такий вид навчання додає ваги зв'язків нейронні мережі.

Нейронні мережі мають нелінійний характер. Через досить потужні процедури оптимізації, протягом великого періоду часу лінійне моделювання являлось основним

методом моделювання. Окрім того, нейронні мережі дозволяють моделювати залежності при наявності великого числа змінних.

Імовірнісна модель навколишнього світу є основою нейромережевих технологій. Подібна модель - основа математичної статистики.

Передбачено багато способів класифікації штучних нейронних систем. Найбільш корисний, з практичної точки зору, класифікаційний ознака полягає в наявності або ненаявності необхідності застосовувати для такої системи навчальну сукупність даних. Якщо навчальна сукупність передбачена, то штучна нейронна система називається заснованої на контрольованій моделі. В іншому випадку модель вважається неконтрольованою.

Нейронні мережі також можуть класифікуватися за такими характеристиками:

- способу з'єднання нейронів;
- застосування обчислень, що виконують нейрони;
- способу передачі шаблонів активності по мережі;
- способу і швидкості навчання.

Нейронні мережі застосовувалися для вирішення практичних завдань усіх типів.

Однією з переваг нейронних мереж є те, що вони дозволяють вирішувати завдання, які виявляються занадто складними для звичайних технологій. Маються на увазі завдання, які не мають алгоритмічного рішення або, для яких алгоритмічне рішення є дуже складним, щоб його можна було визначити аналітично. Власне кажучи, нейронні мережі добре підходять для вирішення завдань, які успішно вирішують люди, але не можуть пояснити, як вони це роблять. [3]

Також однією із переваг нейронної мережі є її простота в використанні. Для навчання нейронної мережі використовують приклади. Ці приклади користувач сам обирає в залежності від очікуваного результату. Хоча користувач виконує якусь роботу у налагодженні нейронної мережі, затрати при повному класичному розрахунку набагато більші чім від використанні нейронної мережі.

Також до основних переваг нейронних мереж як логічного базису алгоритмів слід віднести:

- інваріантність (незмінність, незалежність) методів синтезу нейронних мереж;
- можливість вибору структури нейронних мереж в значному діапазоні параметрів в залежності від складності та специфіки розв'язуваної задачі з метою досягнення необхідної якості рішення.

1.3. Струкура нейронної мережі

Біологічний нейрон імітується в нейронній мережі через активаційну функцію. У задачах класифікації (наприклад визначення спам-повідомлень) активаційна функція повинна мати характеристику «вмикача». Іншими словами, якщо вхід більше, ніж деяке значення, то вихід повинен змінювати стан, наприклад з 0 на 1 або -1 на 1 Це імітує «включення» біологічного нейрона.[4]

При виборі активаційної функціїї зазвичай використовують сигмоїдальну функцію:

$$f(z) = \frac{1}{1 + exp(-z)}$$

Графік даної функції зображений на рисунку 1.1

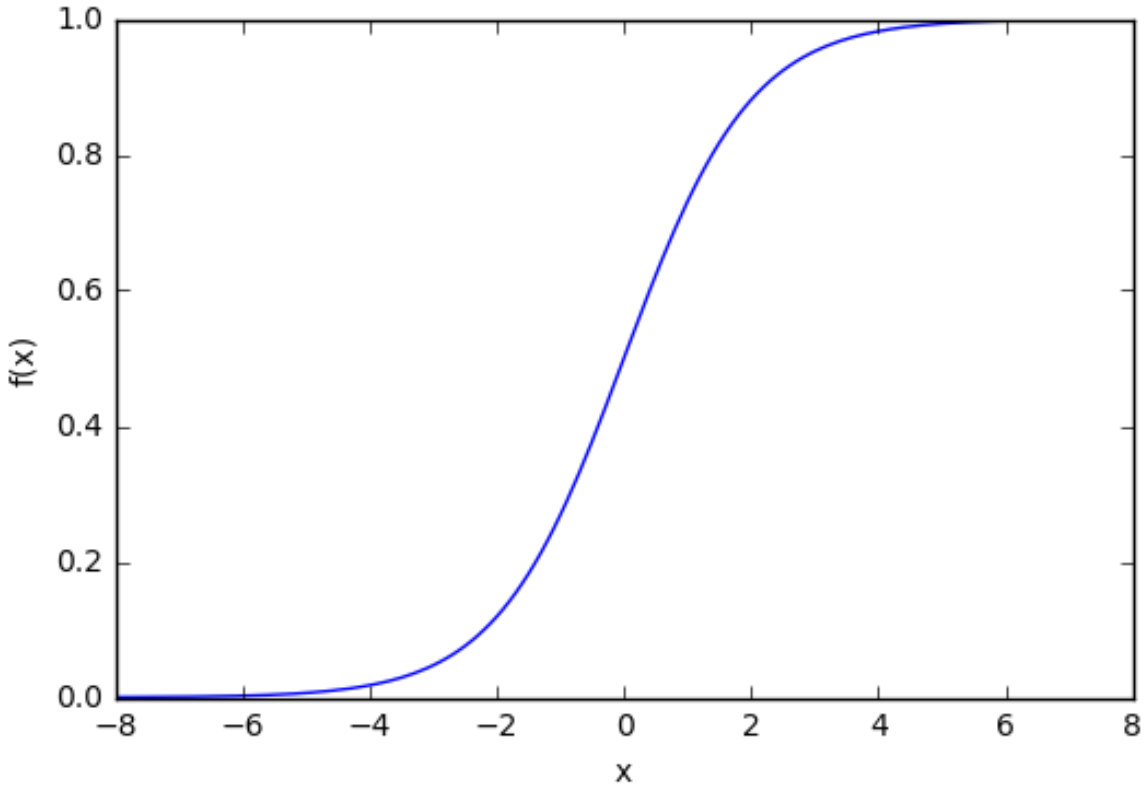

Рисунок 1.1 Сигмоїдальна функція [4]

З графіка можна побачити, що функція «активаційна» - вона зростає з 0 до 1 з кожним збільшенням значення x. Сигмоїдальна функція є гладкою і безперервного. Це означає, що функція має похідну, що в свою чергу є дуже важливим фактором для навчання нейронної мережі.

Як було згадано раніше, біологічні нейрони ієрархічно з'єднані в мережі, де вихід одних нейронів є входом для інших нейронів. Ми можемо уявити такі мережі у вигляді з'єднаних шарів з вузлами. Кожен вузол приймає зважений вхід, активує активаційну функцію для суми входів і генерує вихід.

Кожна нейрона мережа імітує спосіб роботи синапсисів в людському мозку. Традиційне обчислення залучає методику розрахунку для виконання завдання. Вхідна інформація проходить через систему і генерує серію вихідних сигналів. Нейронні мережі ж використовують для обробки даних мережу вузлів, таким чином вона імітує роботу нейронів які пов'язані у синопсис. Приклад синапсису зображений на рисунку 1.2.

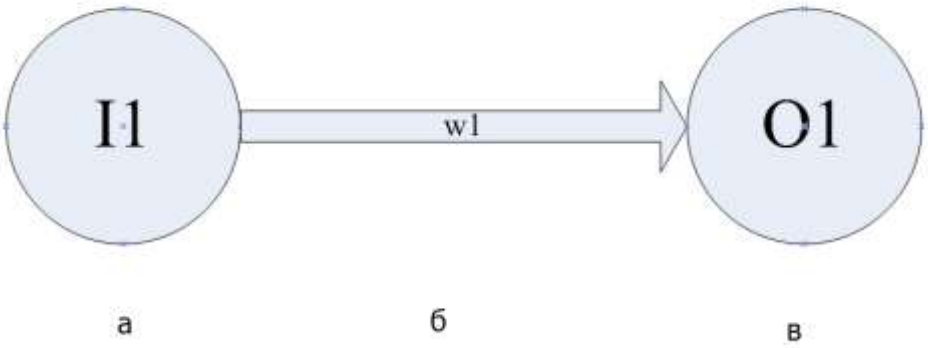

Рисунок 1.2. Схема зв’язку вхідного і вихідного нейрона. а – вхідний нейрон, б – синапсис, в – вихідний нейрон. [5]

Кола на рисунку 1.2 зображують вузли. Вузол є «місцем розташування» активаційної функції, він приймає зважені входи, складає їх, а потім вводить їх в активаційну функцію.

Кожний синапсис має свою вагу. За вагою беруться числа, які потім множаться на вході і підсумовуються в вузлі.

Іншими словами, зважений вхід в вузол має вигляд:

$$x_1 w_1 + x_2 w_2 + x_3 w_3 + b$$

де wi- числові значення ваги, вони є значеннями, які будуть змінюватися протягом процесу навчання. b є вагою елемента зміщення на 1, включення ваги b робить вузол гнучким.

Розглянемо простий нейрон, в якому є по один вхід та 1 вихід:

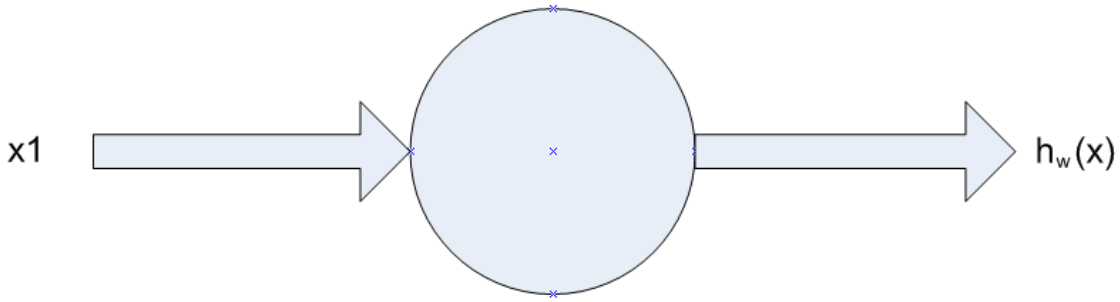

Рисунок 1.3. Нейрон із одним входом та одним виходом

Вхід на функцію активації цього вузла дорівнює $x_1 \cdot w_1$. Зміна функції активації в залежності від зміни коефіцієнту w_1 зображена на рисунку 1.4.

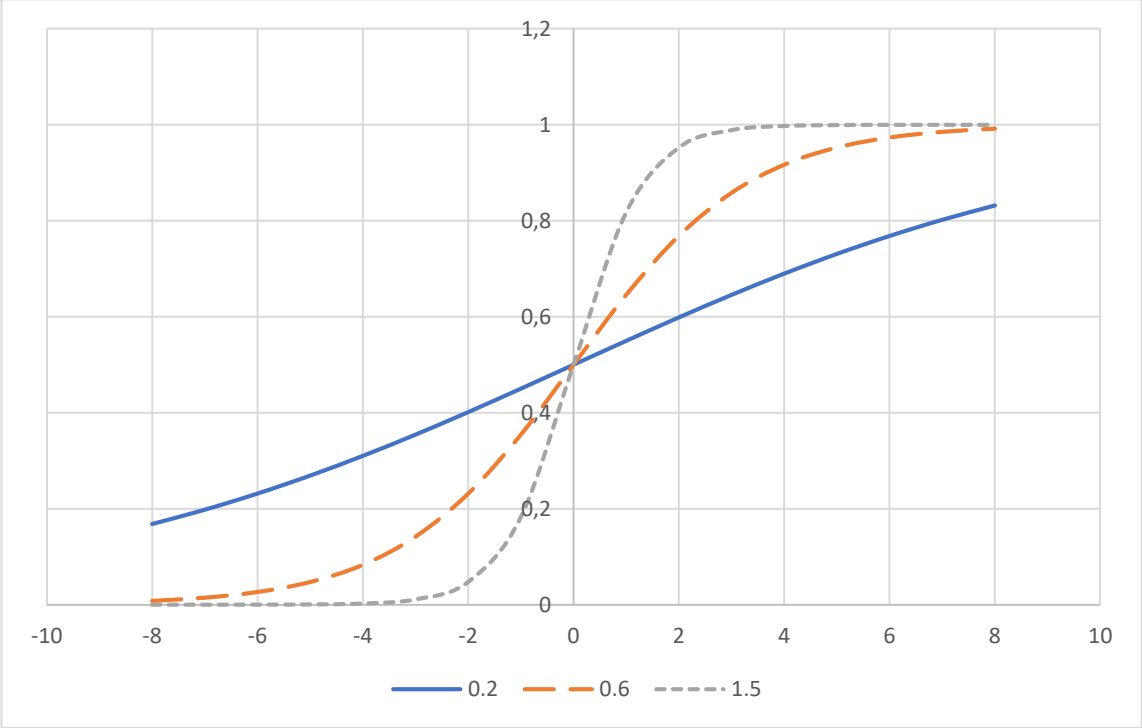

Рисунок 1.4. Зміна функції активації в залежності від зміни коефіцієнту ваги

Тут ми можемо бачити, що при зміні ваги змінюється також рівень нахилу графіка активаційної функції. Це корисно, якщо ми моделюємо різні щільності

взаємозв'язків між входами і виходами. Для того щоб вихід змінювався тільки при x більше 1 потрібно добавити зсув. Розглянемо таку мережу зі зміщенням на вході:

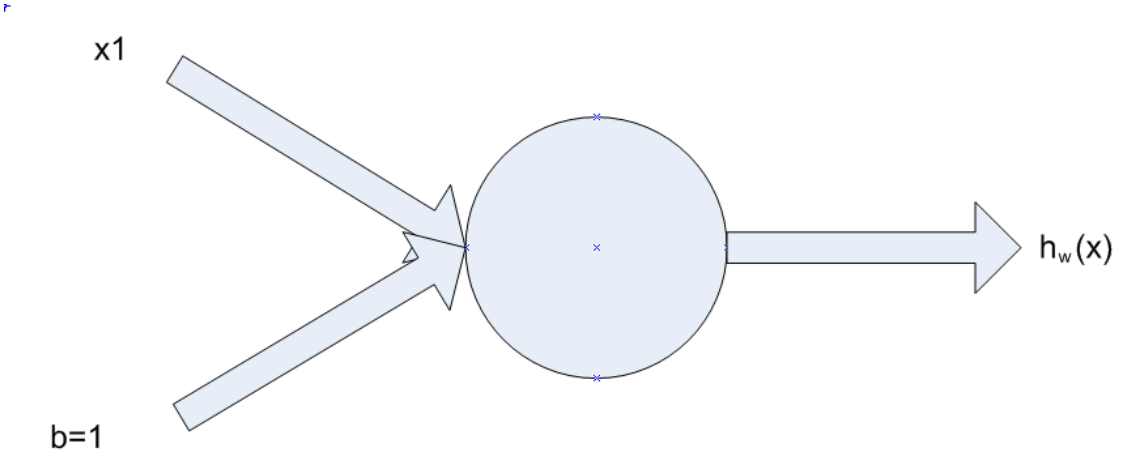

Рисунок 1.5. Нейронна мережа із зміщенням ваги

Графік перетворення функцією активації в такому разу матиме вигляд:

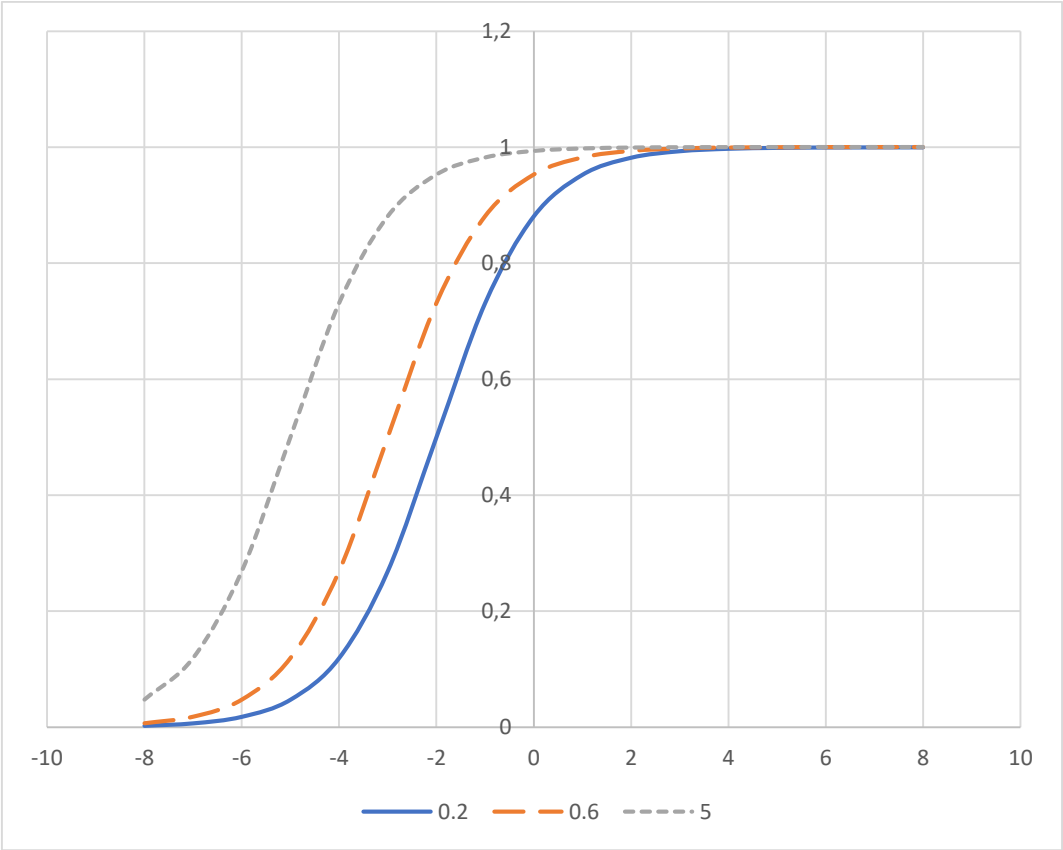

Рисунок 1.6. Зміна функції активації в залежності від зміни коефіцієнту зсуву.

З графіка можна побачити, що змінюючи зміщення b, ми можемо змінювати час запуску вузла. Зсув дуже важливо використовувати у випадках, коли потрібно імітувати умовні відносини.[6]

Найпоширеніша структура нейронної мережі складається з вхідного шару, прихованого шару і вихідного шару. Приклад такої нейронної мережі наведено на рисунку 1.7:

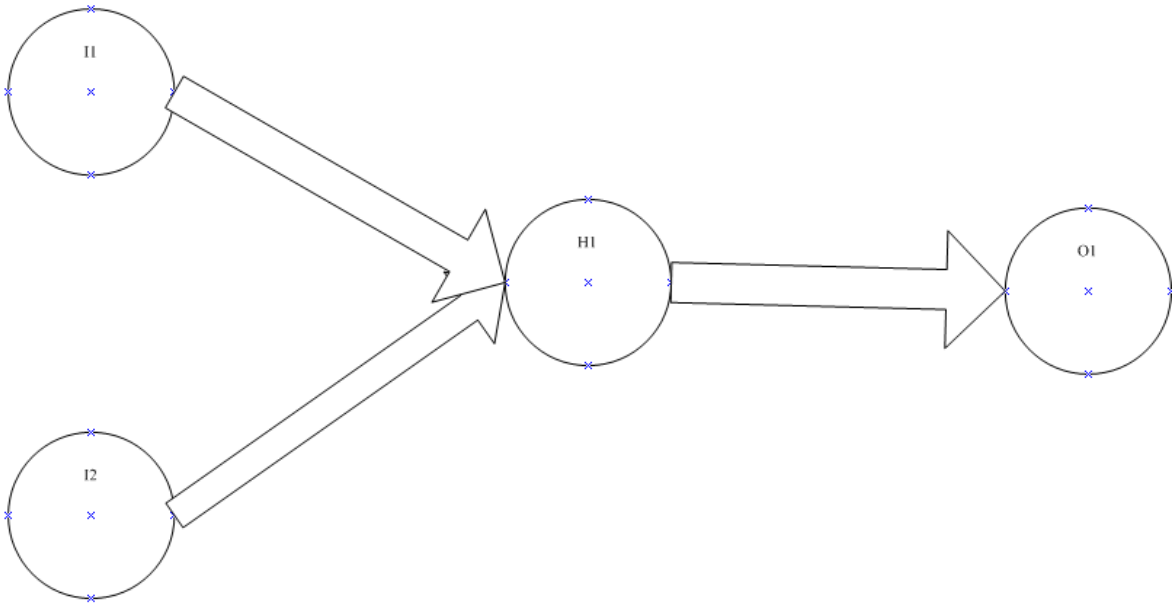

Рисунок 1.7.Структура нейронної мережі яка складається із вхідного, прихованого та вихідного шару.[7]

На рисунку 1.7 I1 та I2-вхідні нейрони, H1 – приховані а O1 – вихідні.

1.4. Нейронна мережа прямого розповсюдження

Найбільш поширеним типом нейронної мережі є мережа прямого розповсюдження. Користь даного типу нейронної мережі є її гнучкість. Тобто існує можливість збільшення точності шляхом збільшення кількості прихованих нейронів, при цьому не змінюючи вагу зв'язків нейронів що використовувались. Тим самим не потрібно робити повторне навчання нейронної мережі. Тому варто створювати

нейронну мережу модульною. Це також спростить підбор необхідної кількості прихованих нейронів для максимально оптимізованої роботи.

Схематичне зображення нейронної мережі прямого розповсюдження зображене на рисунку 1.8.

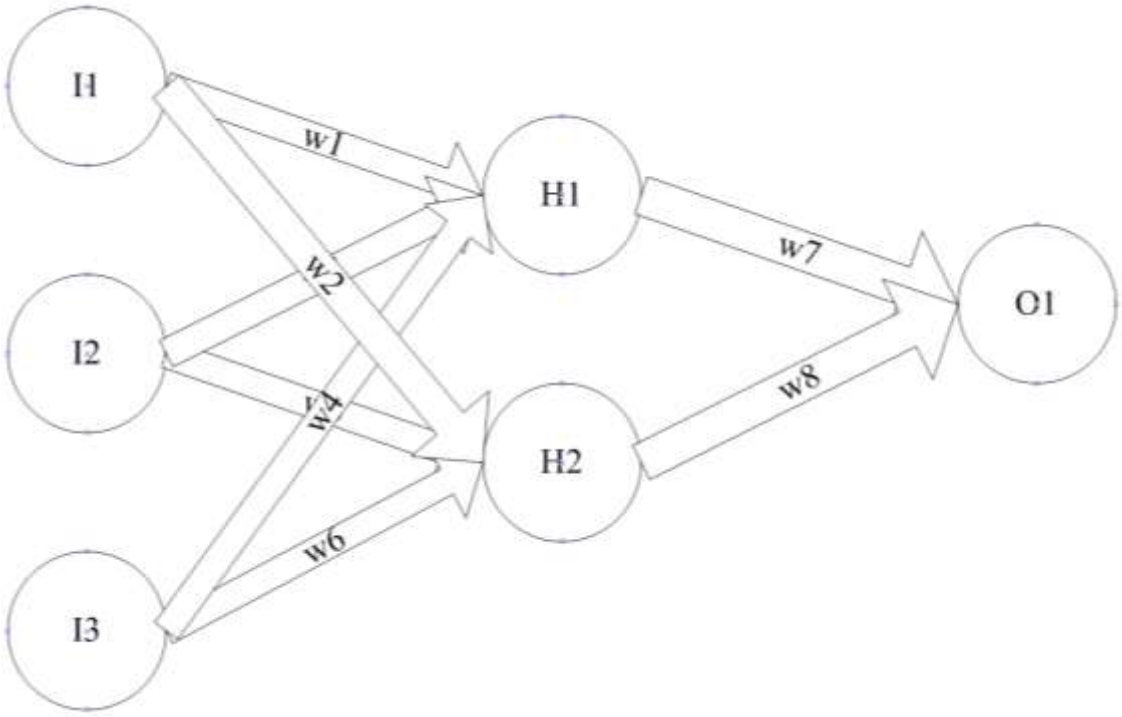

Рисунок 1.8. Схематичне зображення нейронної мережі прямого розповсюдження.

Нейронна мережа прямого розповсюдження вважається найпростішою мережею. Сигнали в такій мережі проходять від входів через приховані елементи та кінець кінцем приходять на вихідні елементи. Основна перевага такого розрахунку це її стійка поведінка. Існують рекурентні нейронні мережі у яких передача сигналу відбувається від кінцевих до початкових нейронів . Такі мережі використовуються тільки в дослідженні нейронних мереж. При розрахунку практичних завдань вони вважаються мало ефективними.

Однією з переваг такої нейронної мережі також є можливість перенавчання в результаті появи нових залежностей в результатах. При цьому основні результати залишаються без суттєвих змін.

Нейрони за структурою своєю організовані в три шари. Вхідний шар служить просто для введення значень вхідних змінних. Кожен з прихованих і вихідних нейронів з'єднаний з усіма елементами попереднього шару за допомогою синапсисів.

При виборі даних між відомими вхідними значеннями та невідомими виходами повинний бути зв'язок. Цей зв'язок може бути змінений шумом, але він повинний існувати.

1.5. Алгоритм вирішення завдання

Нейромережевим алгоритмом вирішення завдань називається обчислювальна процедура, повністю або здебільшого реалізована у вигляді нейронної мережі тієї чи іншої структури (наприклад, багатошарова нейронна мережа з послідовними або перехресними зв'язками між шарами формальних нейронів) з відповідним алгоритмом настройки вагових коефіцієнтів. [8]

Основою розробки нейромережевого алгоритму є системний підхід, при якому процес вирішення завдання представляється як функціонування в часі деякої динамічної системи.

Для її побудови необхідно визначити:

- об'єкт, який виступає в ролі вхідного сигналу нейронної мережі; об'єкт, який виступає в ролі вихідного сигналу нейронної мережі (наприклад, безпосередньо рішення або деяка його характеристика);
- бажаний (необхідний) вихідний сигнал нейронної мережі; структуру нейронної мережі (число шарів, зв'язку між шарами, об'єкти, службовці ваговими коефіцієнтами);
- функцію помилки системи (що характеризує відхилення бажаного вихідного сигналу нейронної мережі від реального вихідного сигналу);
- критерій якості системи і функціонал її оптимізації, що залежить від помилки; значення вагових коефіцієнтів (наприклад, визначаються аналітично

безпосередньо з постановки задачі, за допомогою деяких чисельних методів або процедури налаштування вагових коефіцієнтів нейронної мережі).

Кількість і тип формальних нейронів в шарах, а також число шарів нейронів вибираються виходячи із специфіки вирішуваних завдань і необхідної якості рішення. Нейронна мережа в процесі настройки на вирішення конкретного завдання розглядається як багатовимірна нелінійна система, яка в ітераційному режимі цілеспрямовано шукає оптимальне рішення функції, кількісно визначає якість вирішення поставленого завдання. [9]

Для нейронних мереж, як багатовимірних нелінійних об'єктів, формуються алгоритми настройки вагових коефіцієнтів. Основні етапи дослідження нейронної мережі і побудови алгоритмів настройки (адаптації) їх вагових коефіцієнтів включають:

- дослідження характеристик вхідного сигналу для різних режимів роботи нейронної мережі (вхідним сигналом нейронної мережі є, як правило, вхідні обробляється інформація і вказівку так званого «вчителя» нейронної мережі);
- вибір критеріїв оптимізації (при ймовірнісної моделі зовнішнього світу такими критеріями можуть бути мінімум середньої функції ризику, максимум апостеріорної ймовірності, зокрема при наявності обмежень на окремі складові середньої функції ризику);
- розробку алгоритму пошуку екстремумів функціоналів оптимізації (наприклад, для реалізації алгоритмів пошуку локальних і глобального екстремумів);
- побудова алгоритмів адаптації коефіцієнтів нейронної мережі; аналіз надійності і методів діагностики нейронної мережі та ін.

1.6. Вибір вхідних даних

Вибір початкових умов ітераційної процедури пошуку екстремумів функції є важливим етапом синтезу алгоритмів настройки багатошарових нейронних мереж. Вибір початкових умов повинний бути специфічний для кожного завдання, нейронної

мережі, і бути невід'ємною складовою загальної процедури синтезу алгоритмів настройки багатошарових нейронних мереж. [10]

Якісне рішення цієї задачі в значній мірі може скоротити час налаштування. Складність функціоналу оптимізації зробила необхідної введення процедури вибору початкових умов у вигляді випадкових значень коефіцієнтів з повторенням цієї процедури і процедури налаштування коефіцієнтів.

Ця процедура ще в 1960-і роки здавалася надзвичайно надлишкової з точки зору часу, що витрачається на налаштування коефіцієнтів. Однак, незважаючи на це, вона досить широко застосовується і в даний час.

Для окремих завдань була прийнята ідея вибору початкових умов, специфічних для даної розв'язуваної задачі. Така процедура була відпрацьована для трьох завдань:

- розпізнавання образів;
- кластеризація;
- нейроідентифікація нелінійних динамічних об'єктів.

Системний підхід до побудови алгоритмів пошуку екстремуму функціоналу вторинної оптимізації передбачає в якості одного з режимів перенастроювання коефіцієнтів. Розроблено алгоритми настройки багатошарових нейронних мереж з фільтрацією послідовності значень.

Головне питання - як вибрати структуру багатошарової нейронної мережі для розв'язання обраного конкретного завдання - до сих пір в значній мірі не вирішено. Можна запропонувати лише розумний спрямований перебір варіантів структур з оцінкою їх ефективності в процесі виконання завдання.

Однак оцінка якості роботи алгоритму налаштування на конкретної обраної структурі, конкретному завданні може бути недостатньо коректною. Так як для оцінки якості роботи лінійних динамічних систем управління застосовуються типові вхідні сигнали (ступінчастий, квадратичний та тому подібні). За допомогою реакції нейронної мережі можна отримати її помилку яка слід оцінити.

Подібно до цього, для багатошарових нейронних мереж були розроблені типові вхідні сигнали для перевірки і порівняння працездатності різних алгоритмів

настройки. Природно, що типові вхідні сигнали для таких об'єктів, як багатошарові нейронні мережі, є специфічними для кожної розв'язуваної задачі. В першу чергу були розроблені типові вхідні сигнали для виконання таких завдань. [11]

Основним аксіоматичним принципом застосування нейромережевих технологій замість методів класичної математичної статистики є відмова від формалізованого опису функцій розподілу ймовірностей для вхідних сигналів і прийняття концепції невідомих, складних функцій розподілу. Саме з цієї причини були запропоновані наступні типові вхідні сигнали.

1.7. Висновки

На початку 21 століття однією з основних концепцій розвитку (навчання) багатошарової нейронної мережі є прагнення до збільшення числа шарів, а це передбачає забезпечення вибору структури нейронної мережі, адекватного розв'язування задач, розробку нових методів для формування алгоритмів настроювання коефіцієнтів.

Перевагами нейронних мереж є:

- властивість так званої поступової деградації - при виході з ладу окремих елементів якість роботи системи падає поступово (для порівняння, логічні мережі з елементів І, АБО, НЕ виходять з ладу при порушенні роботи будь-якого елементу мережі);
- підвищена стійкість до зміни параметрів схем (наприклад, вельми значні зміни ваг не призводять до помилок в реалізації простої логічної функції двох змінних).

Широке поширення нейромережевих алгоритмів в області складних та слабких у формулюванні задач привело до створення нового напряму в обчислювальної математики - нейроматематики.

Нейроматематика включає нейромережеві алгоритми вирішення наступних завдань:

- розпізнавання образів;
- оптимізація та екстраполяція функцій;
- теорії графів;
- криптографічні завдання;
- рішення речових систем лінійних і нелінійних рівнянь, звичайних одновимірних і багатовимірних диференціальних рівнянь, диференціальних рівнянь в приватних похідних і ін.

На основі теорії нейронних мереж створено новий розділ сучасної теорії управління складними нелінійними і багатовимірними, багатопов'язними динамічними системами – нейроуправління, що включає методи нейромережевої ідентифікації складних динамічних об'єктів; побудова нейрорегулятора в контурах управління складними динамічними об'єктами і ін.

2 АЛГОРИТМИ НАВЧАННЯ НЕЙРОННОЇ МЕРЕЖІ

Навчити нейрону мережу - значить, обчислити вагові коефіцієнти зв'язку нейронів одного шару з нейронами іншого шару. Алгоритми навчання поділяються на навчання з учителем, без учителя і змішане.

Алгоритм навчання з вчителем передбачає використання еталонних значень в якості вихідних значень мережі; при навчанні без учителя вихідними значеннями є реальні, обчислені при подачі вхідного образу, значення. При змішаному підході частина вагових коефіцієнтів визначається за допомогою навчання з учителем, в той час як інша виходить за допомогою самонавчання.

2.1. Правило навчання Хебба

Одне з перших правил навчання було запропоновано Дональдом Олдінг Хеббом - канадським фізіологом і нейропсихологом. Його також називають одним із творців теорії штучних нейронних мереж. [12]

Правило Хебба свідчить: Якщо при подачі будь-якого способу на вхід системи два і більше нейронів активізується, то коефіцієнт зв'язку між ними посилюється.

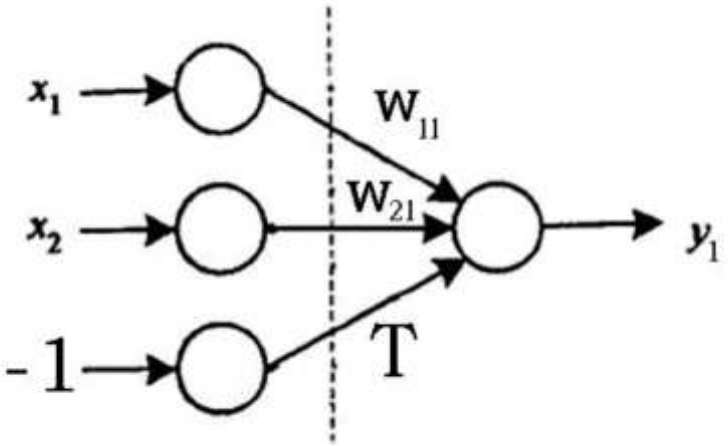

Рисунок 2.1 Одношарова нейронна мережа [13]

Процес навчання зводиться до неодноразової подачі вхідних образів з фіксацією виходу. Відповідно до правила, навчання відбувається шляхом налаштування вагових коефіцієнтів і порогів нейронних елементів.

Використовуючи правило Хебба, можна навчити нейрону мережу реалізовувати різні функції. Це правило використовується як при навчанні з учителем, так і при навчанні без учителя. [14]

Для даної нейронної мережі правило Хебба буде виглядати наступним чином:

$$w_{11}(t+1) = w_{11}(t) + x_1 \cdot y_1$$
$$w_{21}(t+1) = w_{21}(t) + x_2 \cdot y_1$$
$$T(t+1) = T(t) - y_1$$

де T – порог спрацювання нейрону.

При цьому:

$$y_1 = f(S)$$

де $f(S)$ функція активації.

В такому випадку сума буде дорівнювати:

$$S = w_{11}x_1 + w_{21}x_2$$

2.2 Правило навчання Розенблатта

У 1959 р Френк Розенблат - американський вчений, який створив перший нейрокомп'ютер, здатний навчатися на найпростіших завданнях, - запропонував нову модель нейронної мережі, яка отримала назву «персептрон Розенблатта». Схема моделі представлена на наступному рисунку:

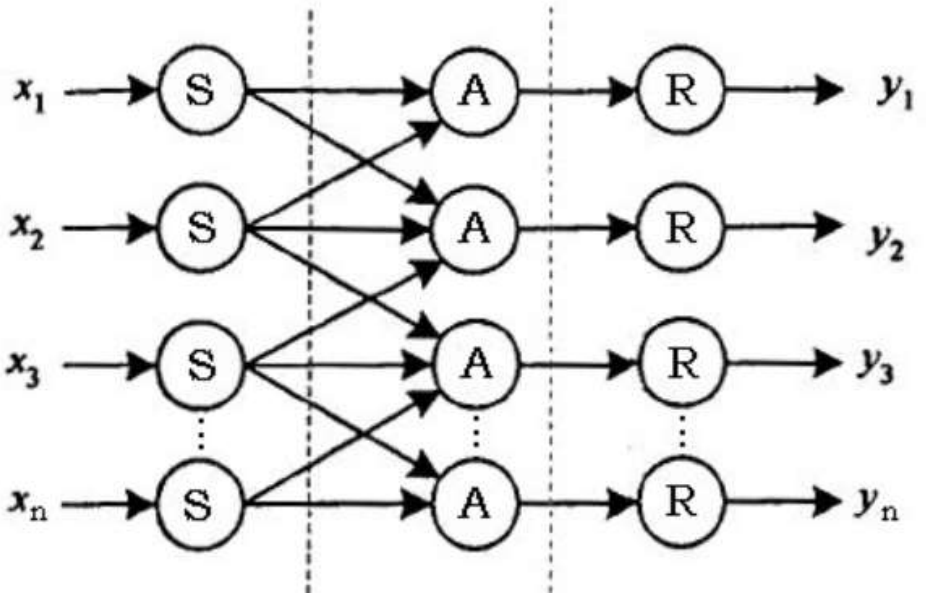

Рисунок 2.2. Персептрон Розенблатта

На данному рисунку зображено наступні нейрони:

- S - сенсорні нейрони;
- A - нейрони, які здійснюють асоціативну обробку інформації;
- R - ефективні нейрони, служать для передачі сигналів збудження.

Формула яка описує навчання персептрон Розенблатта:

$$w_{i,j}(t+1) = w_{i,j}(t) + \alpha * x_i * T_j$$

де α – коефіцієнт яких характеризує швидкість навачання нейронної мережі.

Персептрон складається з набору вхідних і вихідних нейронів з пороговою функцією активації.[15]

Алгоритм навчання складається з наступних кроків:

а) ініціалізація вектора вагових коефіцієнтів відбувається випадковим чином;

б) На вхід мережі подаються вхідні приклади з навчальної вибірки;

в) Якщо відповідь мережі збігається з еталонним значенням, то ваговий коефіцієнт не змінюється. За рахунок цього час навчання персептрона є значно меньшим;

г) якщо ж відповідь мережі не збігається з еталоном, то відбувається модифікація вагового коефіцієнта за формулою навчання;

д) алгоритм виконується до тих пір, поки відповідь мережі не стане дорівнювати еталоному значенню для всіх прикладів або не перестануть змінюватися вагові коефіцієнти.

Відповідно до теорії збіжності, алгоритм навчання сходиться за кілька етапів, якщо існує рішення задачі.

2.3 Правило навчання Уїдроу-Хоффа

У 1960 році професор Стенфордського університету Бернард Уїдроу (Bernard Widrow) разом зі своїм студентом-аспірантом Тедом Хофф (Ted Hoff) на основі правила Хебба розробили нове правило навчання нейронних мереж, назване на честь своїх творців.

Мета навчання за правилом Уїдроу-Хоффа (яке також називають дельта-правилом) є мінімізація середньоквадратичної помилки навчання в заданому просторі вхідних векторів, яка здійснюється за методом градієнтного спуску.

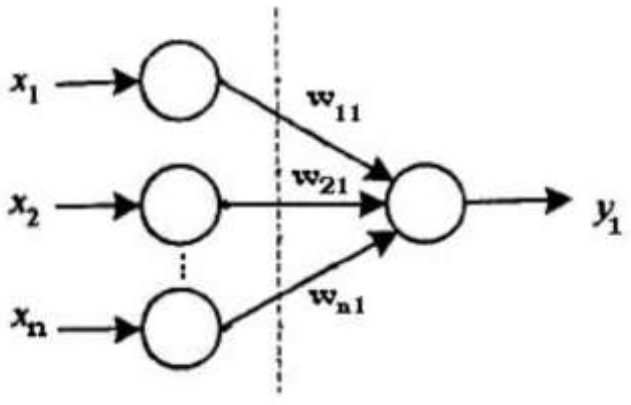

Рисунок 2.3. Одношарова нейронна мережа [16]

Формула, що застосовується при навчанні нейронної мережі, представлена на наступному рисунку:

$$E = \sum_{k=1}^{L} E_k = \frac{1}{2}\sum_{k=1}^{L} (y_1^k - Y^k)^2$$

де E – помилка навчання,

L – кількість вхідних прикладів які входять в навчальн вибірку,

y_1^k – вихідні значення нейронної мережі для k-го прикладу,

Y^k – еталонне значення на виході для k-го прикладу.

$$y_1 = \sum_{j=1}^{n} w_{ji} - T$$

де w_{ji} – знаення вагового коефіцієнту,

T – порог для проходження сигналу на вихід.

Звідси:

$$w_{ji}(t+1) = w_{ji}(t) - \alpha * \frac{dE(k)}{dw_j(t)},$$

$$\frac{dE(k)}{dw_j(t)} = \frac{dE(k)}{dy^k} * \frac{dy^k}{dw_j} = (y^k - Y^k) * x_j^k$$

Таким чином формула зміни вагових коефіцієнтів буде наступною:

$$w_{ji}(t+1) = w_{ji}(t) - \alpha * (y^k - Y^k) * x_j^k$$

де x_j^k – j-й вхід штучної нейронної мережі для k-го прикладу.

Звідси:

$$T(t+1) = T(t) + \alpha * (y_1^k - t^k)$$

де α – коефіцієнт яких характеризує швидкість навачання нейронної мережі,

t^k – порог спрацювання нейрону для k-го пркладу.[17]

В процесі навчання нейронної мережі змінюється ще й поріг спрацьовування.

Формальний алгоритм навчання нейронної мережі відбувається по даному правилу:

а) Вибір і установка коефіцієнтів швидкості і помилки навчання.

б) Ініціалізація вагових коефіцієнтів значеннями, близькими до 0.

в) Подача на вхід нейронної мережі векторів розпізнаваємими прикладами та обчислення вихідних значень.

г) Обчислення різниці між вихідними і еталонними значеннями. Якщо різниця менше помилки навчання, заданої на першому кроці, то процедура завершується, інакше відбувається модифікація вагових коефіцієнтів і порога спрацьовування та перехід до кроку а.

Складність алгоритму полягає у виборі швидкості і помилки навчання. Також велике значення на результат навчання нейронної мережі робить кількість поданих прикладів.

2.4 Навчання нейронної мережі методом зворотного ходу

Навчання такої нейронної мережі реалізується методом зворотного ходу. Тобто якщо отриманий результат нейронної мережі при певному допуску відрізняється від очікуваного результату, взятого із попередніх статистичних даних, то виправлення коефіцієнтів ваги між нейронами починається із кінця нейронної мережі. Вага зв’язків

між нейронами змінюється прямо пропорційно різниці між результатом нейтронної мережі та очікуваним результатом. [18] Таким чином встановлення правильних коефіцієнтів ваги буде найефективніше з початком її навчання. Тому варто на початкових процесах навчання застосовувати усі отриманні дані для опису всіх послідовностей в наслідок яких отримано результати. Схематичне зображення навчання методом зворотного ходу зображено на рисунку 2.4.

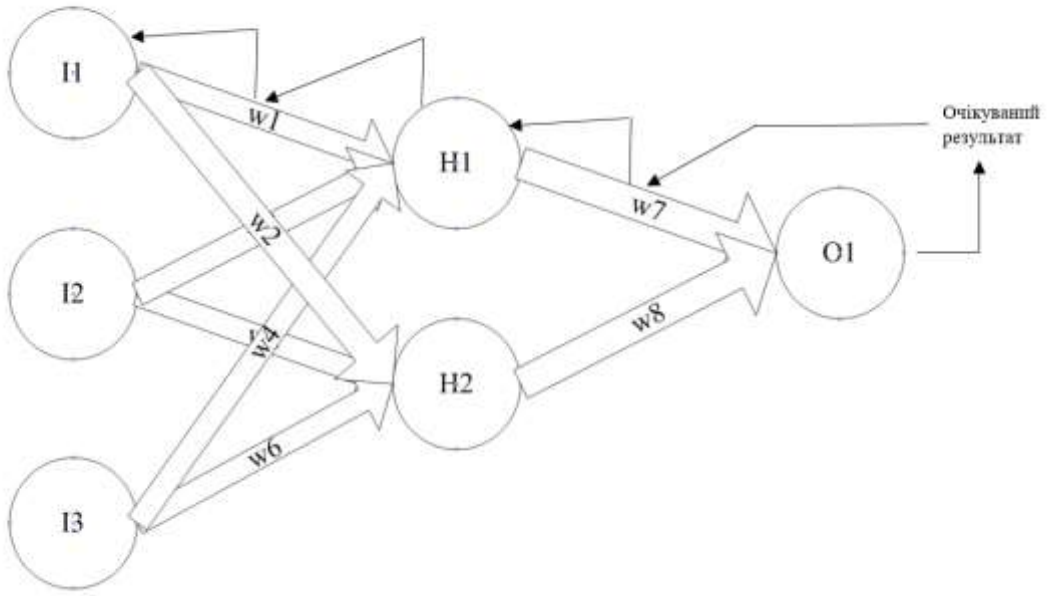

Рисунок 2.4. Схематичне зображення навчання методом зворотного ходу.

Алгоритм навчання складається з декількох основних етапів:

1. Задання ваги нейронів випадковими значеннями або з застосуванням алгоритмів ініціалізації.

2. Розрахунок мережі методом прямого розповсюдження: подаємо на вхід мережі початкові дані і розраховуємо вихідні сигнали нейронів, приховані шари і вихідні шари, підраховуємо різниці між отриманим і очікуваним значеннями на виході, формуємо сигнал помилки.

3. Зворотний хід: поширюємо сигнал на попередні нейрони.

4. Коректуємо ваги нейронів відповідно до правила коригування.

Існують два способи застосування правила коригування - після подачі кожного прикладу навчання (послідовний або інтерактивний, стохастичний режим навчання) і після подачі всього безлічі прикладів (пакетний режим). [20]

Послідовний режим є кращим, так як він більш простий для реалізації і дозволяє подавати приклади у випадковому порядку, що і робить його стохастичним. Послідовний режим ефективний для вирішення складних і великих завдань. Однак зауважимо, що пакетний режим легше зробити паралельним.

2.5 Алгоритм найскорішого спуску

Перспективним в алгоритмі зворотного ходу є вектор градієнта поверхні похибки, який обчислюється по ходу навчання нейронної мережі. Цей вектор вказує напрямок найкоротшого спуску по поверхні з даної точки, таким чином при невеликому зміщенні ваги нейрона похибка зменшиться. [21] Величина кроку зменшується при наближенні до низької точки функції та врешті-решт призведе до мінімуму графіка. Певні труднощі тут представляє питання про те, яку потрібно брати довжину кроків.

При великому значені цього коефіцієнту наближення до результату буде швидким, але існує ризик перестрибнути через рішення якщо поверхня похибки має особливо викривлену форму. Також в такому випадку можливе віддалення від правильного рішення задачі. Приклад великого кроку ваги зображений на рисунку 2.5.

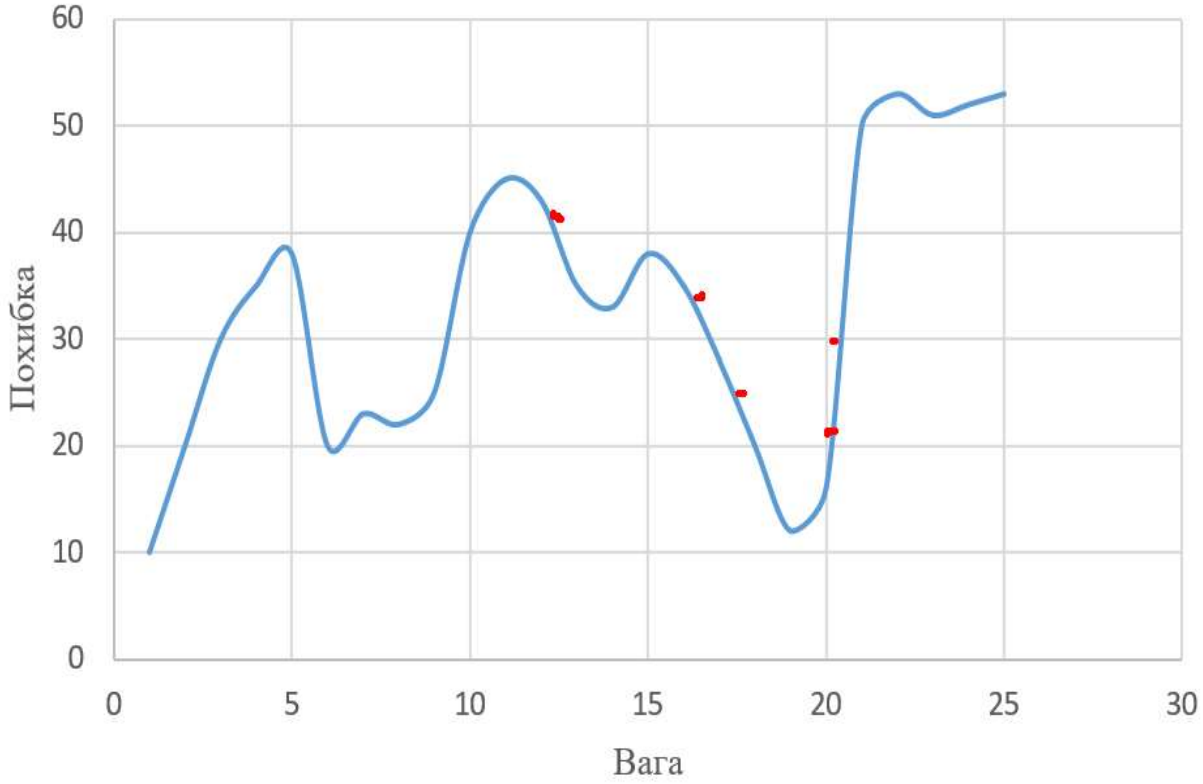

Рисунок 2.5. Зміна похибки від зміни ваги синапсису для великого кроку зміни ваги. [22]

Класичним прикладом такого явища при навчанні нейронної мережі є ситуація, коли алгоритм дуже повільно просувається по вузькому мінімумі функції з крутими схилами, перепригуючи з одного його боку на іншу.

Навпаки, при маленькому кроці, ймовірно, буде схоплено вірний напрям, однак при цьому потрібно дуже багато ітерацій, що істотно зменшує оптимізацію нейронної мережі.

Таким чином коефіцієнт який характеризує зміну кроку переміщення вагів при навчанні потрібно вибирати в залежності від обраної задачі нейронної мережі. Для простих задач із мінімальною кількостями залежностей, вхідних результатів та відповідей, цей коефіцієнт може бут и досить високий.

Приклад малого кроку зображений на рисунку 2.6.

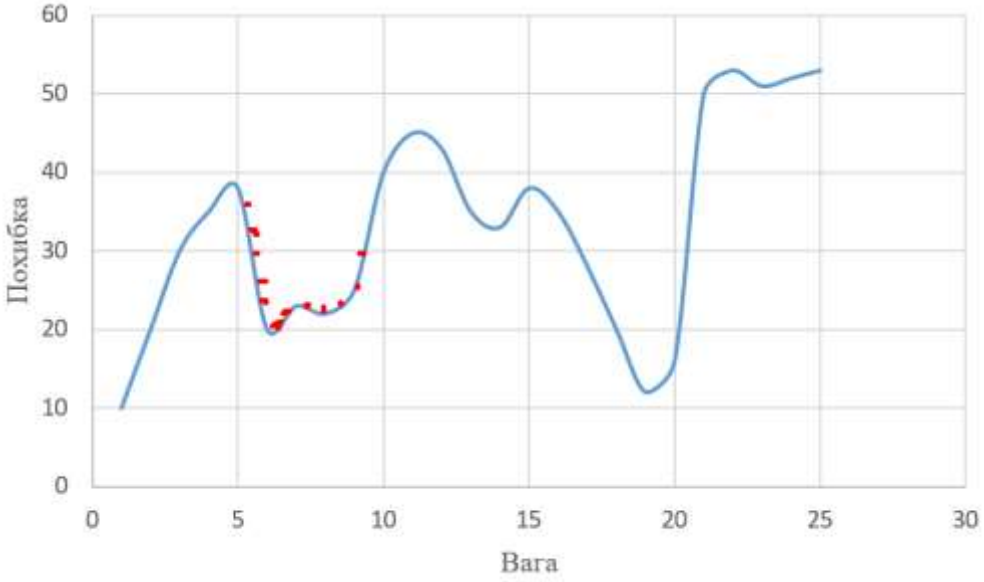

Рисунок 2.6. Зміна похибки від зміни ваги синапсису для малого кроку зміни ваги. [22]

Ці кроки називають епохами. Тобто кожний синапсис містить свій графік розподілу похибки та ваги. Таким чином досить складним завданням є компоновка значень ваги всіх синапсисів щоб похибка була найменшою для всього діапазонна вхідних величин.

Тому для рівномірної настройки нейронної мережі використовують епохи. [23] На кожній епосі на вхід мережі по черзі подаються всі навчальні спостереження, вихідні значення мережі порівнюються з очікуваним результатом і обчислюється помилка. Чім більша помилка при даній вагі тим більший градієнт і тим сильше змінюються ваги. Таким чином відбувається корегування ваги. Це відбувається до тих пір поки загальна похибка не буде відповідати заданій величині.

Складним завданням при виборі бази даних для нейронної мережі є кількість спостережень. Існують певні правила які встановлюють число необхідних спостережень для певної мережі. Найчастіше використовують принцип в якому кількість спостережень має бути в 10 разів більше чім кількість синапсисів між нейронами помножених на кількість вхідних змінних . З ростом кількості змінних, кількість необхідних спостережень зростає нелінійно, так що вже при досить невеликому числі змінних може знадобитися величезне число спостережень.

Не всі вхідні дані представленні у вигляді чисел. Такі дані потрібно інтерпретувати в числові значення, які повинні повністю характеризувати їх зміну. Також деякі вхідні величини, які неспівставні, не варто посилати на вхід однієї нейронної групи. Для такого випадку нейрону мережу поділяють на декілька нейронних груп результати яких встановлюються як вхідні данні інших нейронних груп які розраховують результат нейронної мережі. У такі мережах досить легко виконувати моніторинг розрахунку результату.

За діапазоном величин які приймають вхідні значення потрібно підбирати функцію активації яка нормалізує вхідні данні, тим самим нейрон видає значення в потрібному діапазоні. Для різних груп нейронів використовують різні функції активації.

2.7 Евристичні алгоритми

Крім алгоритмів навчання, що реалізують апробовані методи оптимізації (такі, як алгоритм найшвидшого спуску з моментами, методи змінної метрики, Левенберга-Марквардта або сполучених градієнтів), створено величезну кількість алгоритмів евристичного типу, які надають в основному модифікацію методів найшвидшого спуску або сполучених градієнтів.

Створення евристичних алгоритмів обумовлено повільною збіжністю алгоритму найшвидшого спуску. [24] Як правило, такі методи не мають серйозного теоретичного обґрунтування і засновані на особистому досвіді роботи авторів з нейронними мережами. До найбільш відомих евристичним алгоритмам відносяться Quickprop і Rprop.

Алгоритм QuickProp містить елементи, що запобігають зациклення в точці неглибокого локального мінімуму, виникає в результаті роботи нейрона на фазі насичення сигмоїдальної кривої, де через близькість до нуля похідної функції активації процес навчання практично припиняється.

Вага wij на k-му кроці алгоритму змінюється відповідно до правила коригування:

$$\Delta w_{ij}(k) = -\eta_k \left[\frac{\partial E(w_{ij}(k))}{\partial w_{ij}} + \gamma w_{ij}(k) \right] + \alpha_{ij}^k \Delta w_{it}(k-1)$$

Перший доданок відповідає оригінальним алгоритмом найшвидшого спуску, останнє - фактору моменту, а середній член призначений для мінімізації абсолютних значень ваги. [25]

Коефіцієнт γ призводить до зменшення ваг до відповідних зв'язків.

Також відома спрощена версія алгоритму QuickProp, в якій значення ваг змінюються відповідно. Спрощений алгоритм представлений формулою:

$$\Delta w_{ij}^k = \begin{cases} \alpha_{ij}(k) \Delta w_{ij}(k-1) \text{ для } \Delta w_{ij}(k-1) \neq 0 \\ \eta_0(k) \dfrac{\partial E}{\partial w_{ij}} \end{cases}$$

У ньому зменшено кількість керуючих параметрів.

Простий евристичний алгоритм, який демонструє високу ефективність навчання, - це алгоритм М. Рідміллера і Х. Брауна, званий Rprop.

У цьому алгоритмі при уточненні ваг враховується тільки знак градієнтної складової, а її значення ігнорується. Формула цього алгоритму представлена далі:

$$\Delta w_{ij}(t) = -\eta_{ij}(t) \operatorname{sgn}\left(\frac{\partial E(w_{ij}(t))}{\partial w_{ij}} \right).$$

Коефіцієнт навчання підбирається індивідуально для кожного ваги з урахуванням зміни значення градієнта. Алгоритм Rprop дозволяє значно прискорити процес навчання в тих випадках, коли кут нахилу цільової функції невеликий.

2.8 Алгоритм імітації відпалу

Навчання нейронних мереж, навіть при використанні найефективніших алгоритмів, являє собою трудомісткий процес, далеко не завжди дає очікувані результати. Проблеми виникають через нелінійні функції активації, що утворюють численні локальні мінімуми, до яких не можуть звестися результати нейронної мережі.

На результати навчання величезний вплив справляє підбір початкових значень ваг - ініціалізація нейронної мережі. На жаль, не існує універсального методу підбору ваг, з цієї причини в більшості практичних реалізацій найчастіше застосовується випадковий підбір ваг з рівномірним розподілом значень в заданому інтервалі.

Для підвищення якості роботи нейронної мережі можемо використовувати більш складні алгоритми ініціалізації. Розглянемо алгоритм імітації відпалу для генерування початкових ваг нейронної мережі.

Якщо прийняти енергію стану тіла за цільову функцію, яку необхідно мінімізувати, то високий рівень енергії, що залишився в тілі після швидкої кристалізації, відповідає локальному мінімуму функції.

Досягнення ж глобального мінімуму функції (глобального зменшення енергії тіла) відбувається при поступовому зменшенні температури, хоча в процесі зменшення допускаються ситуації, в яких температура тіла може зростати якийсь час.

У реальних процесах кристалізації твердих тіл температура знижується ступінчастим чином. На кожному рівні вона якийсь час підтримується постійною, що необхідно для забезпечення термічної рівноваги. За рахунок того, що температура

тримається в допустимих межах, що відповідають безперервному зниженню рівня термічної рівноваги, вдається обходити пастки локальних мінімумів.

У разі ініціалізації ваг нейронних мереж за w ми приймаємо набір вагів нейронів, значення цільової функції σ - стандартне відхилення помилки.

Стандартна похибка помилки:

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(y_i - y_{\phi i})^2}$$

Для підрахунку помилки σ – задаємо мережу поточним набором ваг w, потім подаємо один навчальний приклад, здійснюємо один прямий хід по мережі і підраховуємо помилку на виході нейронної мережі.

2.8 Висновки

У даному пункті магістерської дисертації наведено перелік різних методів навчання нейронної мережі. Найбільш поширеним є навчання нейронної мережі методом зворотного ходу.

При виборі методу паралелізації слід звернути увагу на метод навчання даної нейронної мережі. Розпаралелювання не завжди можливе, іноді при фазі навчання нейронні мережі потрібні дані які можливо отримати при розрахунку усіма нейронними мережами даних. Оскільки ці нейронні мережі знаходяться у різних потоках ці результати не будуть розраховані одночасно. В такому випадку створюються точки синхронізації після доходження до яких всіма потоками розрахунок зможе продовжитися. Кількість цих точок повинна бути мінімальною. В залежності від вибору алгоритму навчання можлива різна кількість точок синхронізації, таким чином вибір методу навчання є важливою складовою.

3 АЛГОРИТМИ ПАРАЛЕЛІЗАЦІЇ НЕЙРОННОЇ МЕРЕЖІ

Згідно [26] існує 5 методів паралелізації фази навчання штучної нейронної мережі. Далі наведений список цих методів з коротким описом принципів паралелізації.

При вирішенні задачі за допомогою нейронної мережі зазвичай проводиться безліч експериментів з метою встановлення необхідних параметрів, таких як параметр швидкості навчання і кількість нейронів в різних шарах. З використанням паралелізації фази навчання кілька різних конфігурацій нейронної мережі можуть бути досліджені одночасно. Єдина відмінність між двома фазами навчання складається в параметрі швидкості навчання.

3.1. Паралелізація фази навчання

При використанні паралелізації фази навчання лінійний приріст швидкодії легко досяжний, оскільки зникає необхідність міжпроцесорної взаємодії.

Даний метод пропонує створення копій нейронної мережі. Кількість копій дорівнює кількості процесорів або кількості необхідних потоків. Кожній нейронній мережі на вхід подаються однакові дані, після чого відбувається розрахунок відповідей нейронної мережі та подається відповідь на фазу навчання.

Далі, при фазі навчання, відбувається корегування вагів методом зворотного поширенням помилки. Отриманні дані збираються в масив даних після чого осереднюються.

Схема паралелізації фази навчання представлена на рисунку 3.1.

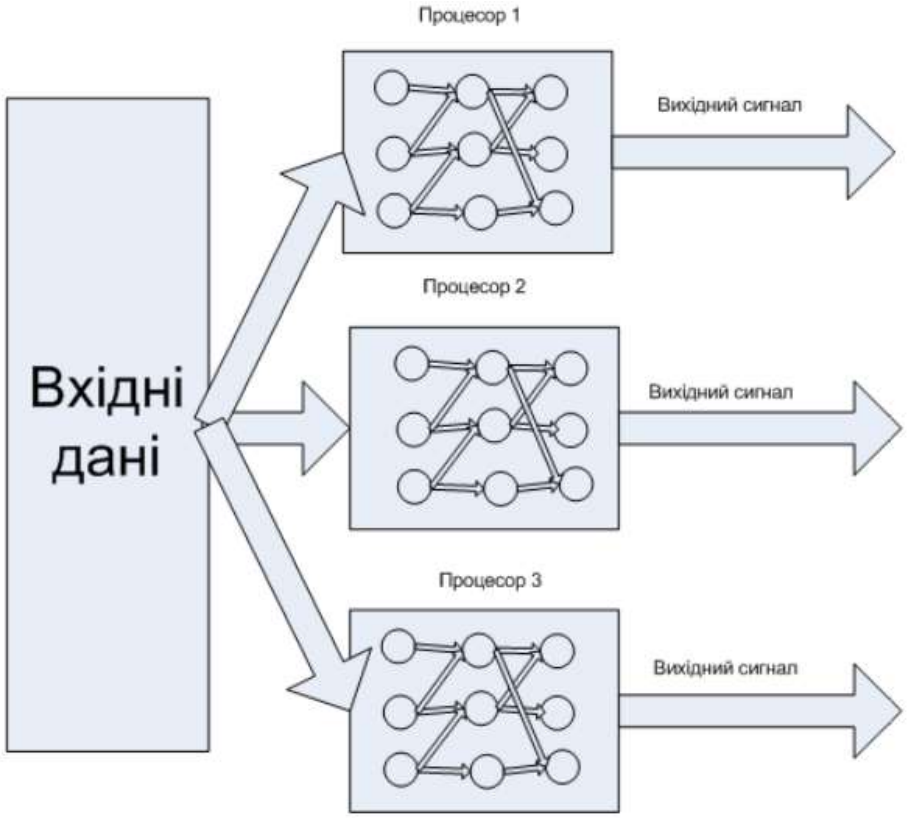

Рисунок 3.1. Схема паралелізації фази навчання

3.2. Паралелізація навчальної вибірки

Часто розмір навчальної вибірки стосовно до задачі, розв'язуваної за допомогою нейронних мереж, може бути досить великим. У однопотоковій системі навчальні вектори будуть спрямовані в мережу по одному.

При використанні паралелізації на рівні навчання вибірки використовують навчання одночасно на декількох різних навчальних вибірках. [27] Це важливо, тому що часто потрібно досить велика кількість навчальних векторів в штучної нейронної мережі для вирішення певного завдання об'ємного розміру.

Особливість даного методу навчання полягає в тому, що на вхід нейронних мереж подаються різні значення тим самими змінюється результати відповіді. Ваги при цьому після їх виправлення відправляються на інші потоки. Після цього відбувається отримання результатів.

Схема паралелізації зображена на рисунку 3.2.

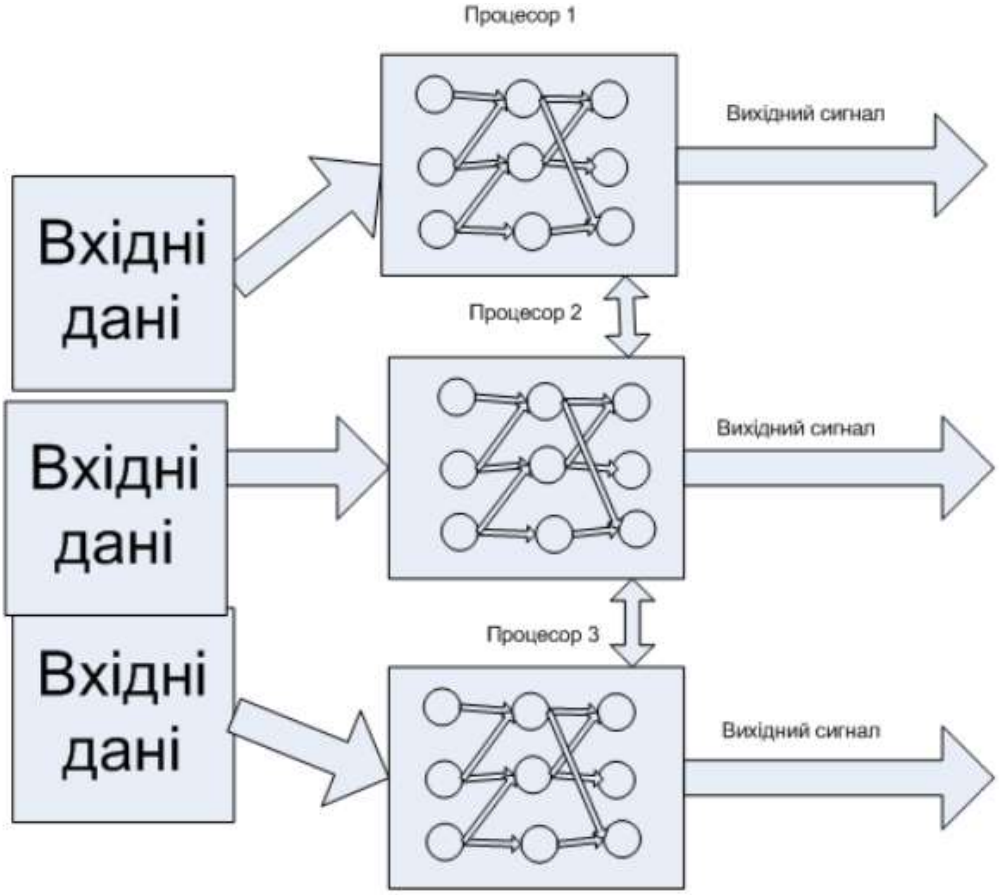

Рисунок 3.2. Паралелізація навчальної вибірки

3.3. Паралелізація на рівні шару.

У таких моделях нейронних мережах як мережах зі зворотнім поширенням помилки, навчальні вектори проходять через мережу за принципом конвеєра. Таким чином, кілька навчальних векторів можуть перебувати в мережі одночасно.

На відміну від перерахованих вище, нейронна мережа не є багатошаровою моделлю, крім випадку, коли поширення навчальних векторів представлено у вигляді шару. [28] Оскільки вхідний шар не виробляє ніяких обчислень, то при розподілі обчислювальних ресурсів вхідного шару не можна досягти ніякого приросту продуктивності.

Схема паралелізації зображена на рисунку 3.3.

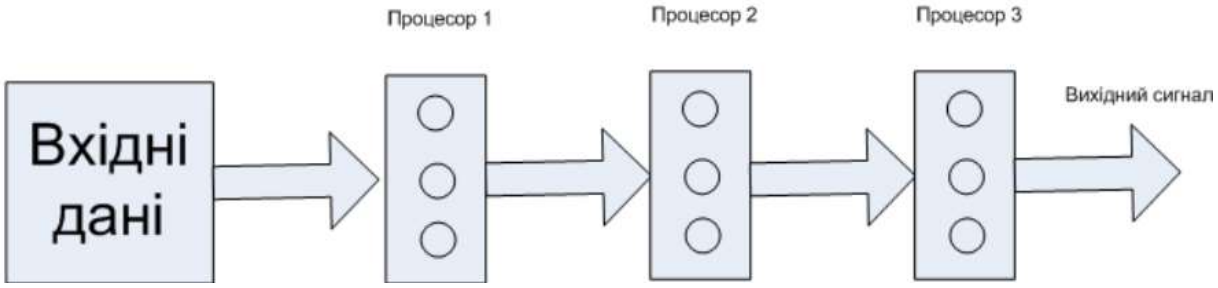

Рисунок 3.3. Паралелізація на рівні шару

3.4. Паралелізація на рівні нейрона

При використанні методу паралелізації на рівні нейронів, нейрон за функціями аналогічний обробляє процесору, так як є також обробляють елементом.

Паралельна обробка як процес на даному рівні розділяє нейрони в рамках одного шару по процесорам, а також при паралельних обчисленнях. В рамках даної системи обробки нейрон або деяке їх кількість співвідноситься з кожним процесором.

Така модель паралелізації є в кожної з моделей штучних нейронних мереж і є одним з популярних методів.

Параллелизация на рівні нейрона є найбільш очевидною з присутніх в моделі нейронної мережі, оскільки нейрон як обробляє елемент нейронної мережі аналогічний окремому процесору.

Паралелізм на рівні нейрона полягає в поділі нейронів (всередині шару, якщо використовується модель з декількома шарами) серед процесорів, і подальшому паралельному обчисленні.

Один або більше нейронів зіставляються з кожним процесором. Паралелізація на рівні нейрона присутня у всіх моделях нейронних мереж, і це найбільш популярний метод в більшості паралельних реалізацій, незалежно від використовуваної моделі нейронної мережі.

Схема паралелізації зображена на рисунку 3.4.

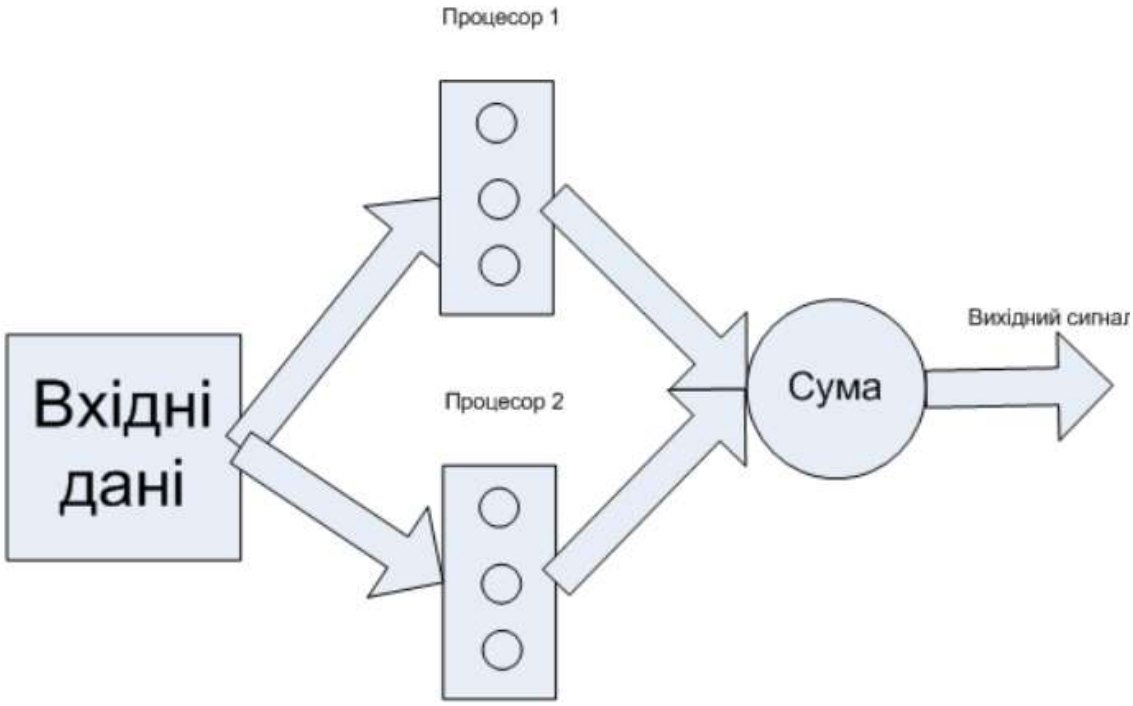

Рисунок 3.4. Паралелізація на рівні нейрона

3.5. Паралелізація на рівні вагів.

Обчислення в межах нейрона також можуть бути розділені між декількома процесорами. Це є дрібномодульний паралелізм і він в першу чергу представлений в апаратних реалізаціях.

Для реалізації паралельності нейронної мережі використовуються як постійне нарощування потужності, так й еволюційні нейрокомпютори, що функціонують на відмінних від фон-неймановского обчислювальних принципах.

Суперкомп'ютери є найбільш поширеним рішенням для реалізацій різних моделей нейронних мереж.

Схема паралелізації зображена на рисунку 3.5.

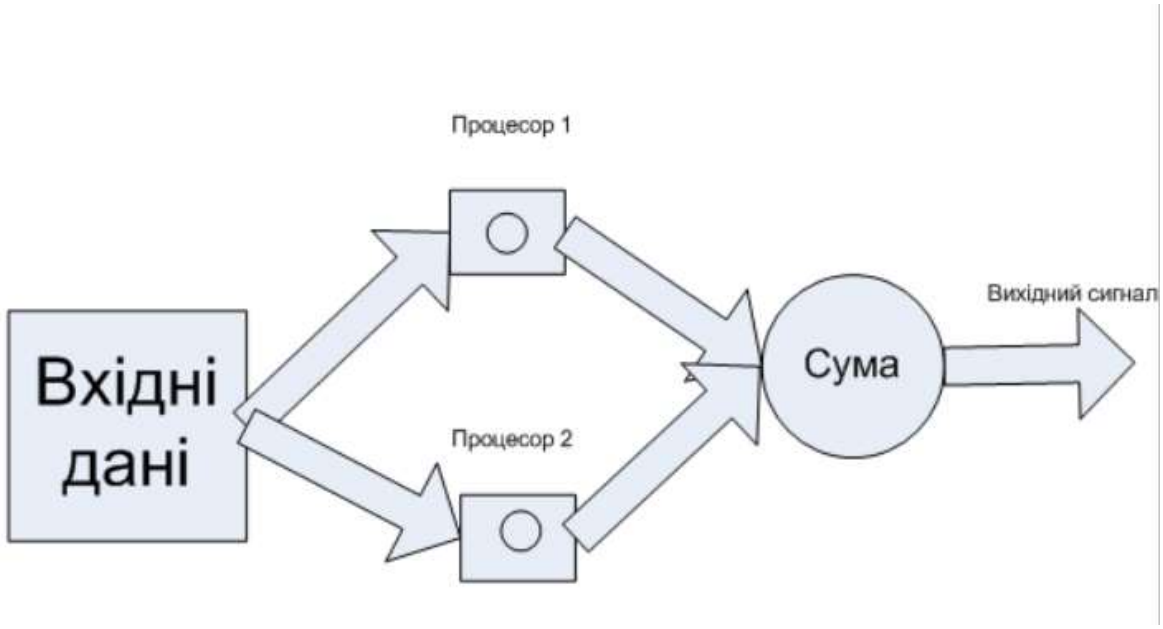

Рисунок 3.5. Паралелізація на рівні вагів

3.5. Висновки

В даному розділі магістерської дисертації наведений перелік основних методів паралелізації процесу навчання нейронної мережі. Вибір методу паралелізації впливає на швидкість навчання та методу навчання.

Метод навчання вибирається в залежності від процесів паралелізації та їх можливостей. При визначенні методу навчання відбувається побудова алгоритму паралелізації. Для цього необхідно розділити алгоритм навчання на етапи між якими існують точки обміну інформацією, так звані точки синхронізації. Для максимально швидкого навчання необхідно щоб цих точок була мінімальна кількість.

Таким чином вибір методу паралелізації тісно пов'язаний із обраним методом навчання.

4 ВИБІР ТА ОПИС ЗАСОБІВ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Процес являє собою автономну середу виконання і може розглядатися в якості програми або програми. Однак сама програма містить декілька процесів всередині себе. Наприклад, програма створена за допомогою Java працює як єдиний процес, який містить різні класи і програми.

Використання різних методів паралелізації призводить до різних швидкостей навчання нейронної мережі. Також це призводить до різних методики навчання, що в свою чергу приводить до різних результатів навчання нейронної мережі. Одним методом можливе наближення до необхідних значень при меншої кількості ітерацій.

Таким чином для паралелізації фази навчання було обрано декілька методів паралелізації, а саме:

- паралелізація навчальної вибірки;
- паралелізація на рівні вагів.

Ці методи мають значні відмінності в структурі та методу навчання, що призводить до збільшення консервативності реалізації методів.

4.1. Вибір бібліотеки розпаралелювання.

Кожна Java програма працює як мінімум з одним потоком - головним потоком. Незважаючи на те, що є дуже багато інших потоків, які працюють у фоновому режимі: управління пам'яттю, управління системою, обробка сигналів і т.д. Але з точки зору нашої програми - головним буде перший потік.

Нить - це два і більше потоків, що виконуються одночасно в одній програмі. Комп'ютер з одноядерним процесором може виконувати тільки один потік, ділячи процесорний час між різними процесами і потоками.

Для різних мов програмування реалізовані різні бібліотеки для створення потоків. У роботі використовується мова програмування Java та пакет її оновлення

Java 8. В даній версії на додачу до класичної бібліотеки Thread також існує бібліотека Stream.

Thread вимагає менше ресурсів для створення і існує в процесі, ділячи з ним ресурси.

Потоки набагато легше процесів, вони вимагають менше часу і ресурсів. Перемикання між потоками набагато швидше, ніж між процесами. Набагато простіше домогтися взаємодії між потоками, ніж між процесами. Класично Java надає два способи програмно створити потік. До них слід віднести:

- реалізацію інтерфейсу java.lang.Runnabl;.
- розширення класу java.lang.Thread.

Функціонально різниці між наведеними вище двома способами немає. Єдиною відмінністю є можливість створення конструктору класу при використанні класу Thread. Якщо необхідний клас надає більше можливостей, ніж просто запускатися у вигляді Thread, то краще реалізувати інтерфейс Runnable. Якщо просто потрібно запустити в окремому потоці, то можна успадковуватися від класу Thread.

Реалізація Runnable є більш кращою, оскільки Java підтримує реалізацію декількох інтерфейсів. Якщо ви успадковуєте клас Thread, то ви вже не можете успадковувати інші класи.

Реалізація інтерфейсу Runnable виконується наступним чином: для того, щоб клас був runnable, ми повинні реалізувати інтерфейс java.lang.Runnable і забезпечити реалізацію методу public void run(). Щоб використовувати цей клас, як потік, ми повинні створити об'єкт Thread, передаючи об'єкт runnable класу, а потім викликати метод start(), щоб виконався метод run() в окремому потоці.

Для застосування класу як потоку необхідну щоб створений клас успадковуваний від класу Thread. В даному класі ми можимо забуспечити реалізацію методу public void run() а також створити конструктор класу. Причому конструктор виконується в основному потоці. Після чого достатньо створити екземпляр класу та визвати метод start(), цим самим паралельно визветься метод run() в середині класу.

В свою чергу Stream API - це підтримка функціонального стилю операцій над потоками, такими як обробка та «згортка» оброблених даних.

У бібліотеках Stream API існує наявність проміжних і кінцевих операцій, які об'єднані в автоматичну форму. Потоки містять джерело (наприклад, колекції тощо) за яким слідують проміжні і кінцеві операції та наводяться їх приклади. Тут варто зауважити, що всі проміжні операції над потоками - ледачі (LAZY). Вони не будуть виконані, поки не буде викликана термінальна (кінцева) операція.

Ще одна цікава особливість, це – наявність методу parallelStream(). Даний метод змінює характеристику визваної дії на паралельне застосування. Ці можливості використовують для поліпшення продуктивності при обробці великих обсягів даних. Паралельні потоки дозволять прискорити виконання деяких видів операці.

Для інтенсифікації багатопотоковості нейронної мережі потрібно провести оптимізаційні тести бібліотек Thread та Stream, та обрати найбільш оптимальну.

Для цього був розроблений тестовий програмний продукт який дає нагрузку на кожний потік однакову, та проведений дослід залежності часу на виконання завдання та кількісті потоків. Для розрахунку використовувався процесор AMD A10-5745M APU with Radeon(tm) HD Graphics, частотою 2.1 GHz. Кількість ядер даного процесору 4.

У відповідності із дослідом був побудований графік даної залежності. Він зображений на рисунку 4.1.

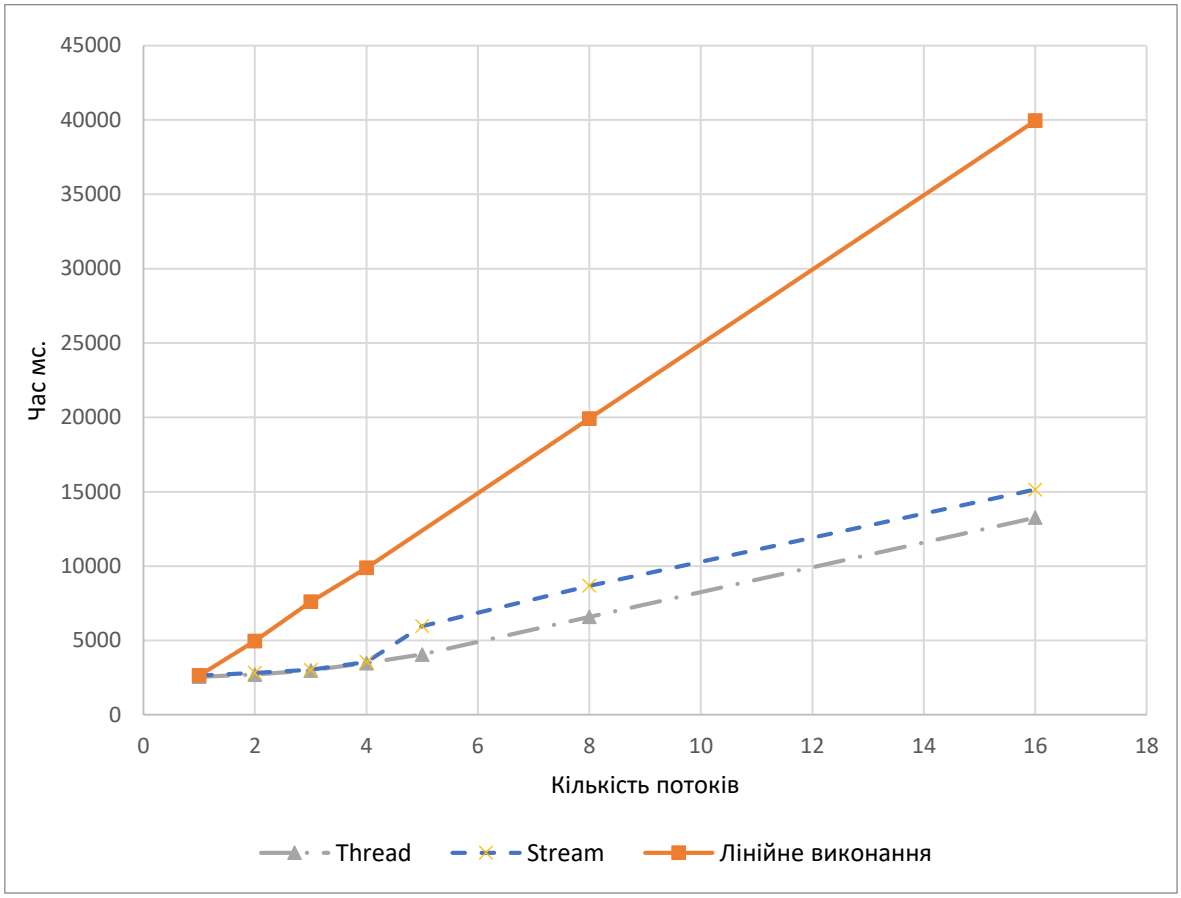

Рисунок 4.1 Залежність часу виконання завдання від кількості потоків для різних методів багатопотоковості.

На графіку зображене лінійне виконання задач, тобто який повинний бути результат без використання багатопотоковості. Порівнюючи криву для бібліотек Thread та Stream видно, що для 1-4 потоків результат між ними однаковий, але оскільки моделювання використовувалось на 4-х ядерному процесорі то при використанні 5 та більше потоків система використовує час для зв'язку між потоками.

Різні бібліотеки використовують різні алгоритми для переключення між потоками, таким чином існує різниця. Тому за допомогою отриманих даних для моделювання багатопотоковості буде використовуватись клас Thread.

4.2. Задачі нейронної мережі

Навчання нейронної мережі можна поділити на два етапи. Першим етапом являється розрахунок відповіді нейронної мережі методом прямого ходу. На даному етапі використовуються ваги які були створенні при попередній ітерації або при створенні нейронної мережі. Отримана відповідь в подальшому використовується в другому етапі, таким чином другий етап не може бути розпочатий без першого.

В другому етапі отримавши відповідь нейронної мережі та порівнявши її із еталонним результатом визначається помилка нейронної мережі. Ця помилка впливає на ваги попереднього шару нейронів. Таким чином навчання нейронної мережі відбувається методом зворотного ходу.

Для перевірки працездатності нейронної мережі була створена тестова нейронна мережа основним завданням якої було визначення правильного значення вимірювальної величини. В даному випадку вимірюється кількість частинок які потрапляють в детектор.

Діапазон вимірювання в детекторі буде знаходитися від 100 до 350 штук. Оскільки функція активації для даної нейронної мережі вибрана сигмоїда то вхідні дані необхідно конвертувати по пропорції від 0 до 1. Такі значення були обрані через результат функції активації, цей результат також лежить в межах від 0 до 1.

Конвертування буде відбуватися за формулою:

$$i = \frac{1}{(350 - 100)} \cdot (n - 100);$$

де i – вхідні дані (0…1);

n – кількість частинок, шт (100…350).

Усі сконвертовані числа представленні на рисунку 4.2.

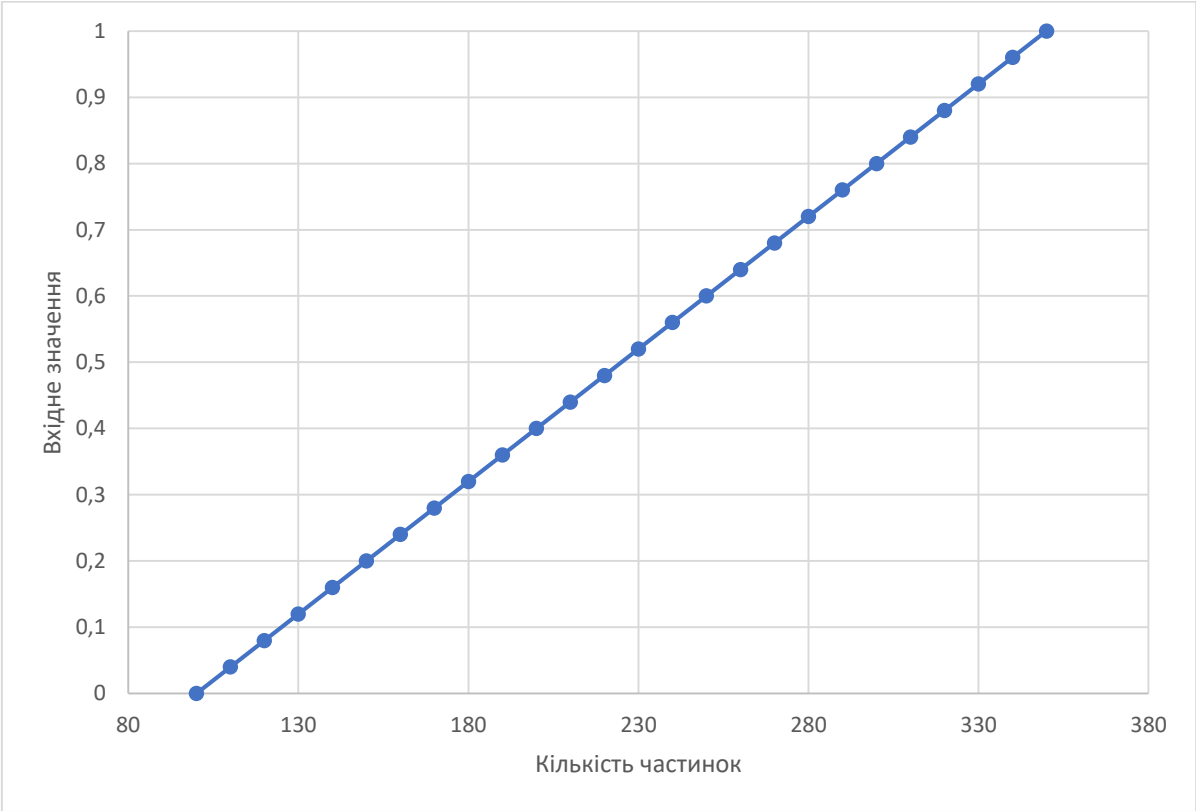

Рисунок 4.2 Конвертація чисел для входу у нейронну мережу.

Якщо кількість частинок мала то вони вступають в взаємодію із іншими частинками і активізують їх. Таким чином це означає, що на детектор потрапить більше частинок ніж насправді об’єкт випромінює. А для великої кількості частинок цей ефект вносить зворотній характер. [29] Калібровка детектора в цих випадках являється досить складним процесом. Потрібно вибрати середнє значення для максимально точного вимірювання. Функція яка описує похибку детектора в залежності від кількості частинок які випромінює об’єкт складно описуєма.

Різниця між вимірюваними частинками та тими що вилітають представлена на рисунку 4.3.

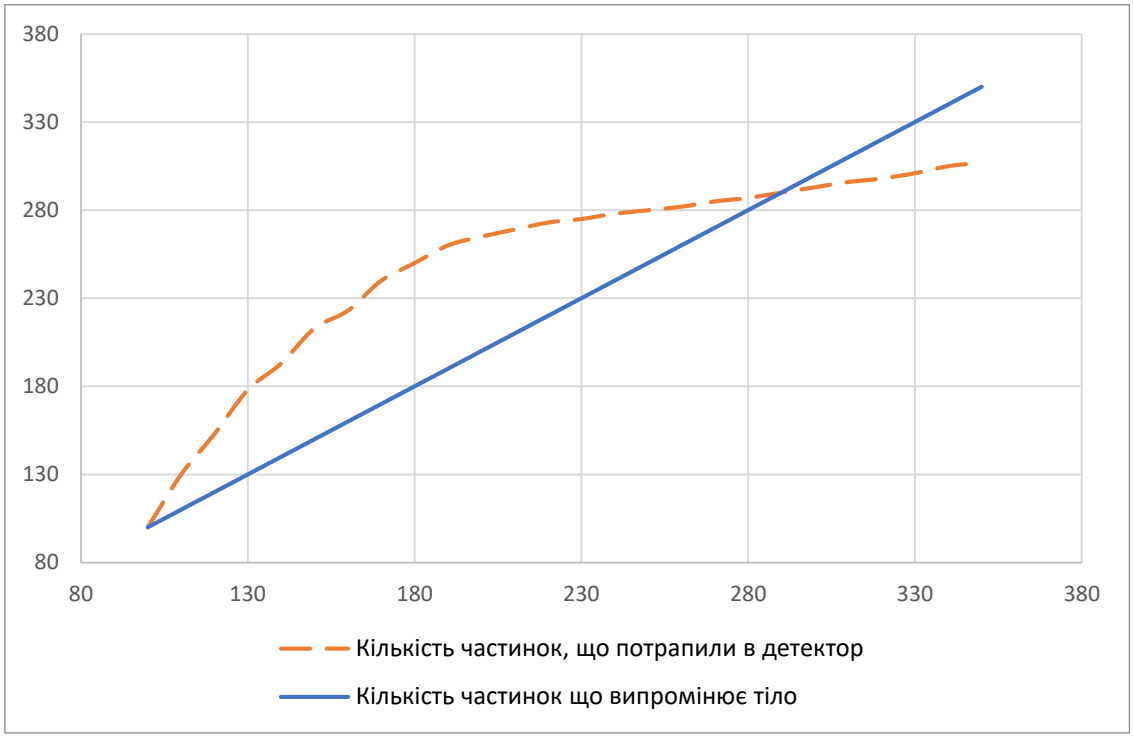

Рисунок 4.3 Кількість частинок які випромінює тіло та поглинаються детектором.

Дані які поступають на вхід нейронної мережі та очікуванні результати представленні на рисунку 4.4.

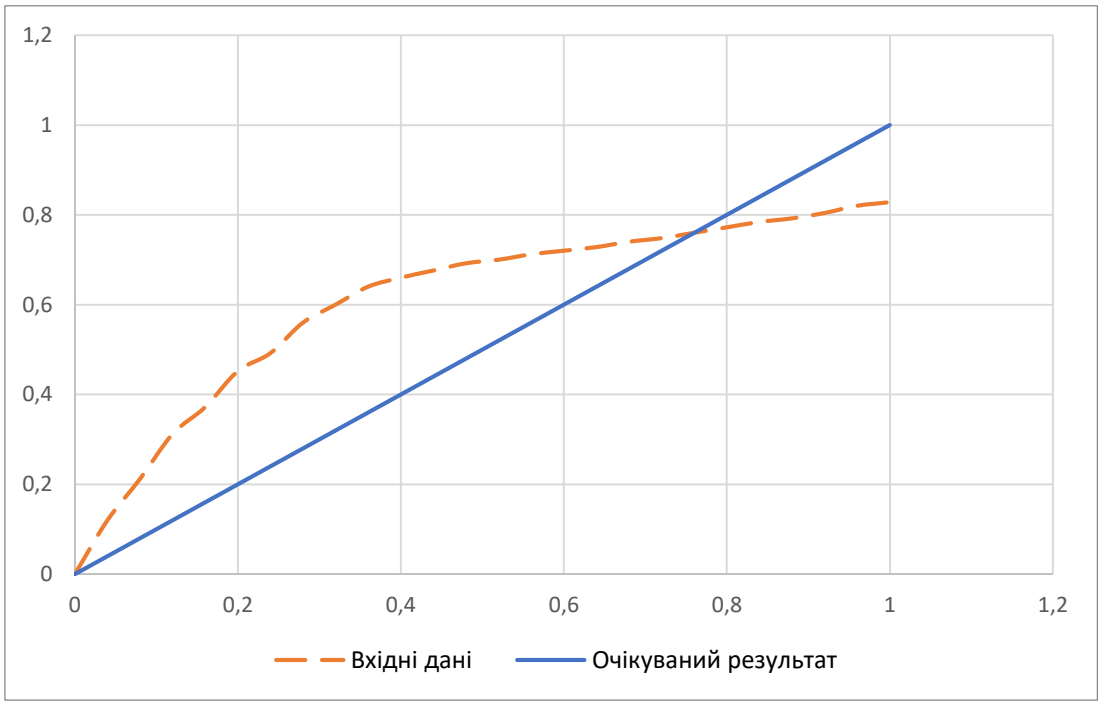

Рисунок 4.4 Сконвертовані дані нейронної мережі

Далі наведена таблиця яка показує конвертацію даних для навчання нейронної мережі:

Таблиця 4.1 Дані для навчання нейронної мережі

Дослідні значення, шт		Сконвертованні значення	
Кількість частинок що випромінює тіло	Кількість частинок, що потрапили в детектор	Еталонний результат	Вхідні значення
1	2	3	4
100	100	0	0
110	130	0.04	0.12
120	153	0.08	0.212
130	178	0.12	0.312
140	193	0.16	0.372
150	213	0.2	0.452
160	223	0.24	0.492
170	240	0.28	0.56
180	250	0.32	0.6
190	260	0.36	0.64
200	265	0.4	0.66
210	269	0.44	0.676
220	273	0.48	0.692
230	275	0.52	0.7
240	278	0.56	0.712
250	280	0.6	0.72
260	282	0.64	0.728
270	285	0.68	0.74
280	287	0.72	0.748

Продовження таблиці 4.1

1	2	3	4
290	290	0.76	0.76
300	293	0.8	0.772
310	296	0.84	0.784
320	298	0.88	0.792
330	301	0.92	0.804
340	305	0.96	0.82
350	307	1	0.828

Відхилення виміряних даних від тих що випромінює тіло зображено на рисунку 4.5.

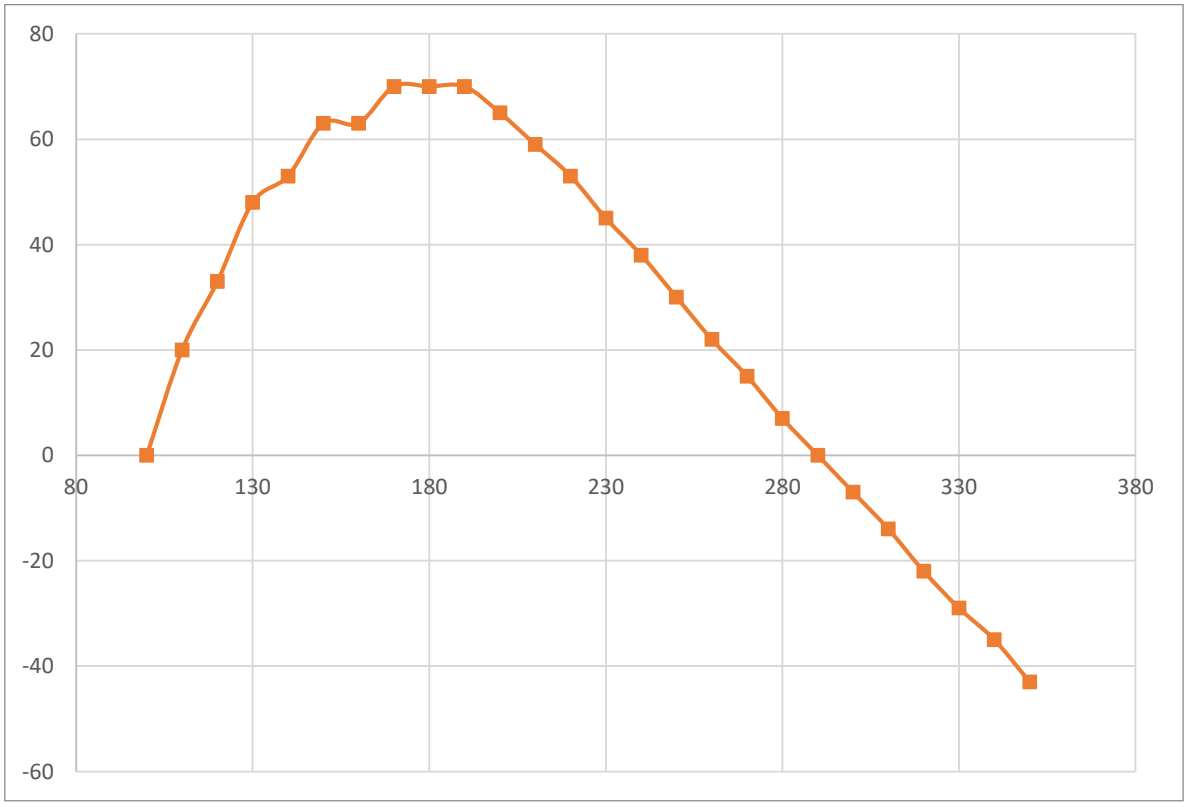

Рисунок 4.5 Похибка вимірювальних значень

Сконвертовані значення для нейронної мережі зображенні на рисунку 4.6.

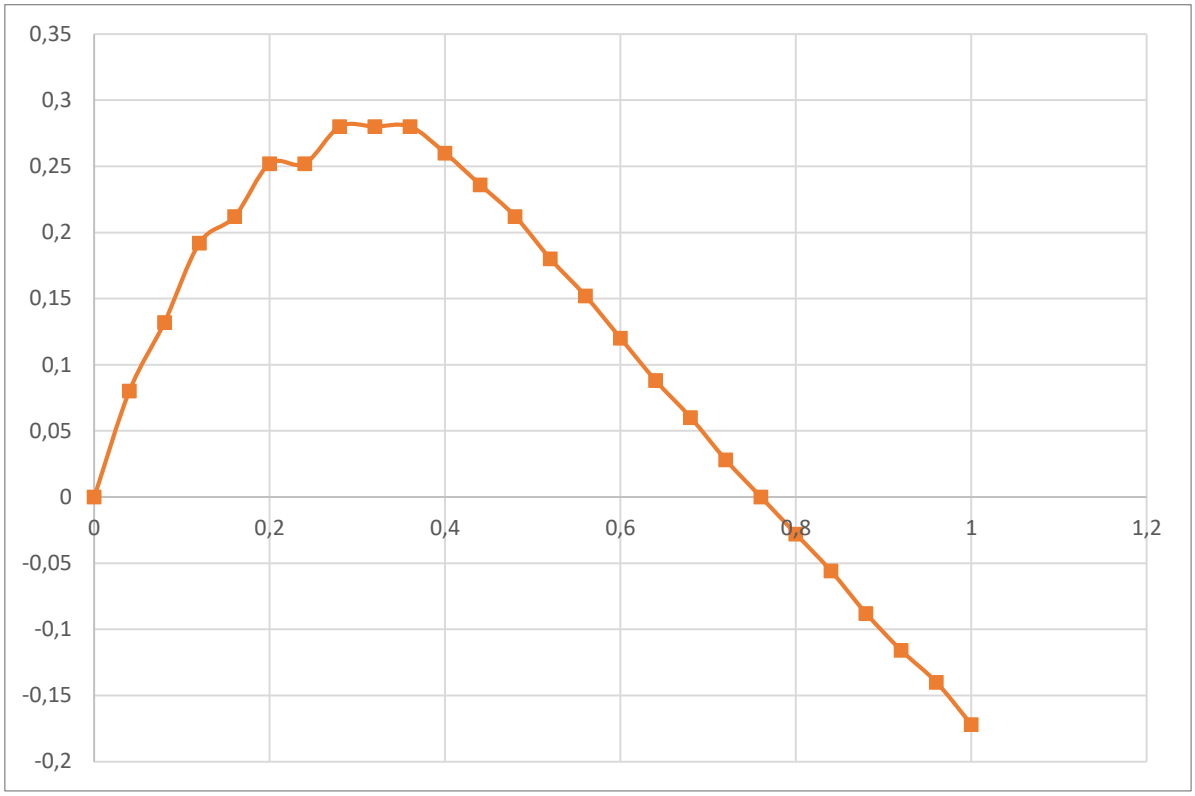

Рисунок 4.6 Сконвертована похибка вимірювальних значень

Загальна сумарна похибка для всіх значень детектора дорівнює 974 шт., якщо сконвертувати то отримаємо 3.896. Розділивши дане число на кількість вимірювань отримаємо середню похибку, вона дорівнює 37,46 шт., та для сконвертованого числа 0,1498, що дорівнює 14,98%. Максимальна похибка вимірювання складає 70 шт., що в відсотковому співвідношенні дорівнює 28%.

Задача нейронної мережі полягає у визначенні кількості частинок які випромінює тіло маючи на вході дані детектора. Успішним результатом навчання вважається якщо середня похибка нейронної мережі буде менша ніж у детектора.

4.3 Алгоритм навчання нейронної мережі

Для навчання нейронної мережі був обраний алгоритм зворотного розповсюдження помилки. Для початку необхідно знайти помилку нейронної мережі. Для цього потрібно щоб нейронна мережа порахувала вхідні дані та видала результат.

Розрахунок буде поділений на етапи який дорівнює кількості не розрахованих шарів нейронів. [30] До них відносяться скриті нейрони та вихідні нейрони. При створенні нейронної мережі усі ваги синапсисів заповнюються випадковими числами в межах від 0 до 1.

Спочатку для кожного нейронна сумують добуток вагів та сигналів нейронів на шар нище, що з'єднуються за допомогою синапсисів. Ця сума представлена формулою:

$$N = w_1 \cdot U_1 + w_2 \cdot U_2 + ... + w_n \cdot U_n;$$

де N значення на вході в нейрон,

w_n – вага синапсису між нейроном n попереднього шару та нейроном, що розраховується,

U_n – сигнал від нейрона попереднього шару.

Для отримання значення яке нейрон буде передавати наступним шарам, потрібно до отриманої суми застосувати функцію активації, в даному випадку сигмоїду. Формула вихідного сигналу нейрона:

$$N' = \frac{1}{1 + e^{-N}};$$

де N' – сигнал на виході нейрона.

Даний сигнал отримують наступні шари. Цей розрахунок в повній мірі підходить і для розрахунку вихідного нейрону. Сигнал який виводить скритий нейрон є відповіддю нейронної мережі.

Другим етапом являється виправлення вагових коефіцієнтів. Якщо відповідь нейронної мережі менша ніж очікуваний результат вагові коефіцієнти будуть знижені. Якщо відповідь більша та вагові коефіцієнти збільшаться. Зміна цих коефіцієнтів залежить від сигналу нейронів які пов'язують їх синапсиси. Таким чином якщо один із вхідних сигналів дорівнює 0 то зміна вагових коефіцієнтів

синапсисів, які сполучають вхідний нейрон із нейронами наступного шару, не буде відбуватися.

Формула для розрахунку помилки нейронної мережі буде наступною:

$$err = a - exp;$$

де err - помилка нейронної мережі,

a – результат розрахунку нейронної мережі,

exp – очікуваний результат.

Далі необхідно найти дельта вагів за формулою:

$$w\_d = err \cdot a(1 - a);$$

де w_d – дельта вагів,

вираз a(1 – a) – похідна функції активації.

Дельта вагів характеризує необхідну швидкість зміни вагів.

Після цього необхідно виправити значення вагів за формулою:

$$w_i = w_i' - (U_i \cdot w\_d \cdot lr);$$

де w_i – виправлене значення вагів,

w_i’ – старе значення вагів,

U_i – сигнал який видає нейрон який пов’язаний з синапсисом ваги якого розраховуються.

lr - learning rate – коефіцієнт яких характеризує величину зміни вагів. Підбирається користувачем в залежності від складності залежностей даних у нейронній мережі.

4.4 Висновки

У даному пункті магістерської дисертації були розглянуті основні засоби програмної реалізації. Були розглянуті основні методи паралелізації нейронної мережі які використовуються в програмному продукті, до них слід віднести:

- паралелізація навчальної вибірки;
- паралелізація на рівні вагів.

Також були проаналізовані дві бібліотеки Thread та Stream API на швидкість опрацювання даних. Був створений тестовий програмний продукт для визначення найбільш оптимальної бібліотеки. Через швидше опрацювання тестової задачі для багатопотоковості був вибраний клас Thread.

Була створена тестова модель завдання для нейронної мережі з опис вхідних даних та очікуваного результату. Задачею нейронної мережі являється визначення кількості частинок які випромінює тіло маючи на вході дані детектора. При цьому успішним результатом навчання вважається якщо середня похибка нейронної мережі буде менша ніж у детектора.

Був представлений алгоритм навчання нейронної мережі. Даний алгоритм поділяється на два етапи: етап розрахунку відповідей нейронної мережі та етап перевизначення вагів нейронної мережі.

5 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Для даної роботи була вибрана нейронна мережа прямого розповсюдження. Також було прийнято рішення створити дану мережу модульною, тобто існує можливість міняти кількість вхідних сигналів, скритих нейронів, рядів скритих нейронів та вихідних сигналів без суттєвих затрат. Модульність нейронної мережі робить її більш гнучкою, таким чином даний програмний продукт може виконувати безліч задач.

Для створення нейронної мережі виконувалась технологія об’єктно орієнтованого програмування у повному обсязі. Синапсиси та результат розрахунку нейронної мережі записуються в окремі текстові файли (Connection.txt та Result.txt), що робить роботу нейронної мережі стійкою до збоїв. Також це дозволяє зберігати результат навчання нейронної мережі на будь який час.

5.1 Структура нейронної мережі

Нейрони із яких складається нейронна мережа характеризуються окремим класом NeuronUnit. У нейронній мережі він представляє з себе масив. Його основні параметри це індинтифікатор та його величина. Індинтифікатор характеризує тип нейрона (вхідний, скритий та вихідний), його номер та для скритих нейронів ряд у якому він розташований.

Для синапсисів між нейронами існує окремий клас Connection. Він містить два індинтифікатора які вказують на нейрони які з’єднанні, а також вагу цього з’єднання.

Загальний клас Neuron характеризує собою нейронну мережу. Він представляє з себе сукупність вхідних скритих та вихідних нейронів, а також їх зв’язки між собою. Даний клас містить в собі методи для розрахунку результату нейронної мережі та проведення навчання її методом зворотного розповсюдження помилки. Тобто даний клас має всі інструменти та параметри для створення нейронної мережі.

Оскільки даний програмний продукт записує та зчитує дані із текстових файлів створенні класи GetDate та WriteDate, які конвертують отриману інформацію в текстову або в змінні в залежності від обраного класу. В класі GetDate описується метод отримання вагових коефіцієнтів для синапсисів (GetConnection()) та навчальну вибірку (GetKnow()).

Навчальна вибірка задається окремим текстовим файлом який користувач повинний обрати у графічном інтерфейсі програми. Кожна навчальна вибірка записується окремою стрічкою там має наступний вигляд: «[Вхідне число] : [Очікуваний результат]»

Конструктор класу Neuron приймає на вхід назву нейронної мережі, від неї залежить місце знаходження файлів нейронної мережі, та самі нейрони. Ці дані конструктор записує в параметри класу та генерує класи Connection для кожного синапсису між нейронами.

При створені класу Connection виконується провірка на існування вагів між нейронами, якщо нейрон створюється вперше то ваги генеруються випадково і записуються в файл Connection.txt.

5.2 Навчання нейронної мережі

В класі Neuron існує два методи які виконують процес навчання. Метод Calculate() розраховує відповідь нейронної мережі. Для можливості виконання даного методу необхідно щоб для вхідних нейронів були присвоєні вхідні значення. Другий метод WriteError(double Actual), який виконує функцію корегування вагів нейронної мережі. На вхід функція приймає очікуваний результат.

Метод Calculate() для розрахунку використовує два етапи: розрахунок для скритих нейронів та для вихідних нейронів. За допомогою циклів відбувається прохід по кожному нейрону в відповідному ряді. Після чого відбувається перехід на наступний скритий шар нейронів. Потім обраний нейрон перевіряється на існування за допомогою порівняння HidenNeuron[H_row][H] != null, де H_row – ряд або шар

скритого нейрона а Н порядковий номер нейрону в шарі. Ця провірка є необхідною оскільки існує можливість нестворених нейронів в масиві, через двовимірність масиву.

Блок-схеми даного алгоритму представлена на рисунку 5.1.

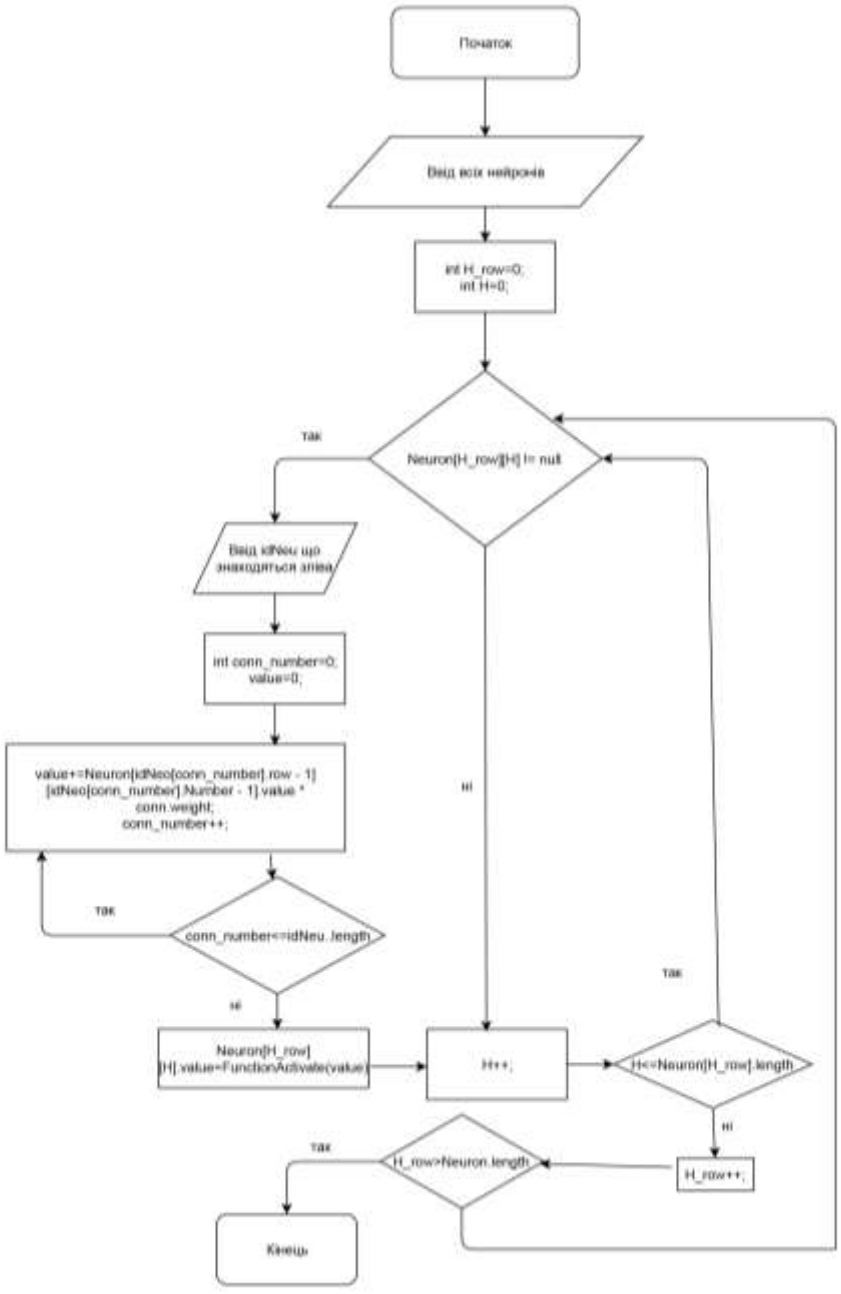

Рисунок 5.1 Блок-схема алгоритму розрахунку відповіді нейронної мережі

Потім за допомогою класу GetDate відбувається отримання ідентифікаторів нейронів, що знаходяться зліва або в попередньому шарі від даного нейрона. Далі

виконується цикл який сумує добутки вагів та сигналів нейронів які з’єднані з ліва. Для першого прихованого ряду є виключення в розрахунку суми, а саме у випадку вибору сигналу від нейронів зліва вибирається вхідні нейрони. У інших випадках сигнал береться із прихованих нейронів.

Блок-схеми другої частини алгоритму представлена на рисунку 5.2.

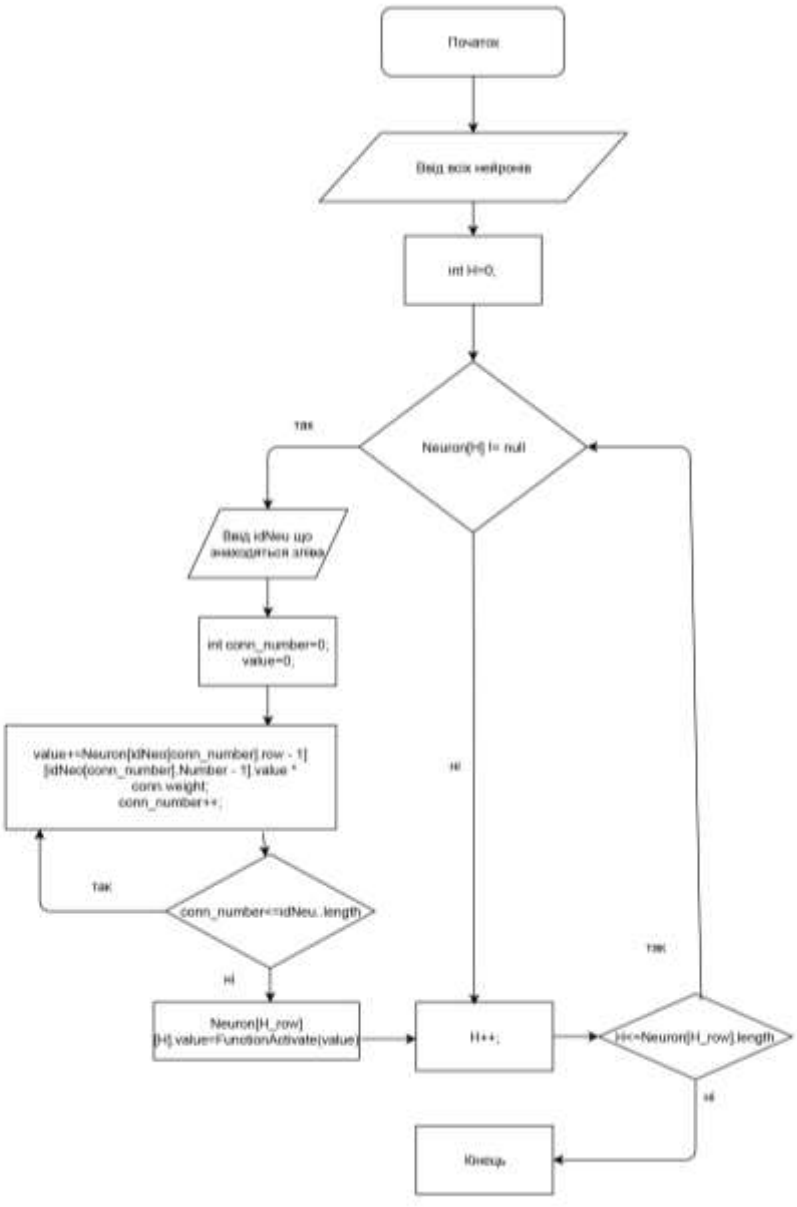

Рисунок 5.2 Друга частина блоку-схеми алгоритму розрахунку відповіді нейронної мережі

Для вихідних нейронів алгоритм подібний. Єдина відмінність полягає у відсутності рядів, оскільки вихідні нейрони не містять в собі шарів. За допомогою

циклів відбувається прохід по кожному нейрону. Потім за допомогою класу GetDate відбувається отримання ідентифікаторів нейронів, що знаходяться зліва або в попередньому шарі від даного нейрона. Далі виконується цикл який сумує добутки вагів та сигналів нейронів які з'єднані з ліва.

У якості функції активації була вибрана сигмоїда. Дана функція описує величини із різним знаком. Її діапазон вхідних чисел змінюється від -6 до 6. Такий малий діапазон чисел спрощую роботу нейронної мережі, за рахунок простіших дій над числами.

Другий метод WriteError(double Actual) має за мету корегування вагів нейронної мережі. Отримане число відповіді використовується для визначення похибки нейронної мережі. Алгоритм методу поділяється на дві частини. Перша розраховує ваги які сполучають вихідні нейрони та скриті. Друга розраховує ваги між скритими нейронами і між скритими та вхідними.

У першій частині застосовується цикл для перерахунку всіх вихідних нейронів. Для кожного вихідного нейрону розраховується помилка та дельта вагів. Після чого знаходяться нейрони які знаходяться зліва від даного нейрону. Потім створюється цикл який перераховує зв'язки даного нейрону та тих що знаходяться зліва. В циклі відбувається корегування вагів кожного синапсису який з'єднаний із даним нейроном.

Для другої частини спочатку запускається цикл який перераховує всі ряди скритих нейронів. В середині циклу запускається ще один який перераховує всі нейрони які знаходяться в даному ряді. Потім відбувається перевірка на існування даного нейрону. Якщо нейрон існує то відбувається пошук нейронові як знаходяться справа від даного. Це потрібно для розрахунку помилки та дельта вагів даного нейрону. Після знаходження цих чисел відбувається пошук нейронів які знаходяться зліва від даного. Запускається цикл який перераховує всі зв'язки даного нейрону з нейронами зліва та для кожного з них виконує корегування. На цьому всі ваги нейронної мережі скореговані.

5.3 Розпаралелювання нейронної мережі

Розпаралелювання даної нейронної мережі відбувається двома способами. Для розпаралелювання навчальної вибірки кількість епох ділять на кількість потоків. Ці потоки створюються за допомогою класу MyStream який успадкував клас Thread. Для конролю вхідних значень та значень які отримуються від даних потоків створений клас control. Таким чином кожний потік буде проходити свою навчальну вибірку. Вхідні значення в такому випадку різні. В кожному потоці запускається метод Calculate() та метод WriteError(double Actual). Всі отримані ваги відправляються в клас control. Таким чином існує масив значень вагів для одних і тих самих синапсисів, після чого для кожного синапсису відбувається усереднення вагів нейронної мережі. В даному випадку усереднення вагів призводить до зміни розрахункового алгоритму нейронної мережі, що означає що питання використання даного алгоритму розпаралелювання потрібно досліджувати в подальшому.

Розпаралелювання вагів відбувається за допомогою створення класів StrWeightKnow для кожного синапсису. Ці класи успадковані від класу Thread, що означає що вони є паралельними. В даних потока виконується лише визначення корегування вагів нейрона. Таким чином ефект прискорення нейронної мережі є незначним. Для управління такими потоками використовується клас ContKnow який виконує методи Calculate() та частково WriteError(double Actual), окрім частини коректировки вагів, в головному потоці.

5.4 Інтерфейс програмного забезпечення

Для інтерфейсу використовувалась стандартна бібліотека javax.swing. Інтерфейс програмного продукту зображений на рисунку 5.3.

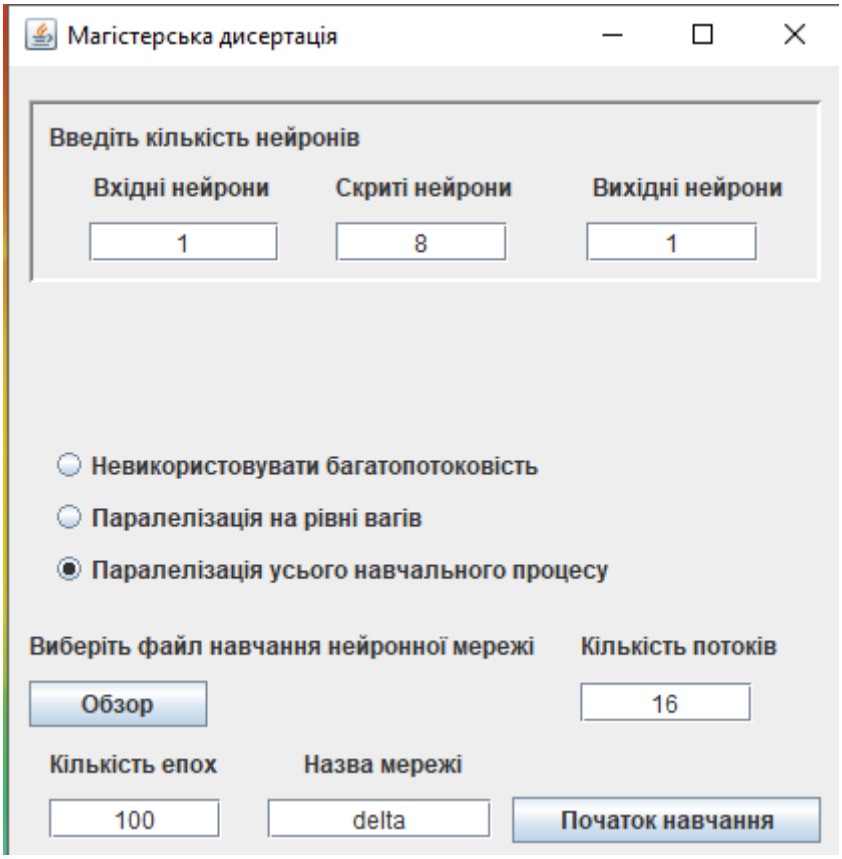

Рисунок 5.3 Інтерфейс програмного продуту

А рисунку 5.3 зверху зображено 3 поля, вони призначенні для визначення кількості нейронів відповідної групи. Нижче розташований CheckBox, якщо даний елемент активований то при навчання нейронної мережі буде застосований метод багатопотоковості.

Для вибору файла навчаннянейронної мережі існує кнопка обзор, яка відкриє діалогове вікн із вибором текстового документу із навчальною вибіркою.

Також інтерфейс програми дозволяє обрати кількість епох в нижньому лівому краю інтерфейсу. Існує також поле для вибору назви нейронної мережі. Воно знаходиться знизу посередині інтерфейсу. Після натискання кнопки старт відбувається перевірка коректності заповнення всіх даних, якщо все вірно то відбувається запуск навчання нейронної мережі.

5.5 Налаштавання нейронної мережі

Для перевірки програмного продукту була створена нейронна мережа, яка складалась із 1 вхідного, 10 скритих, які знаходились в одному ряді, та 1 вихідного.

Коефіцієнт LearningRate дорівнює 0,5. Усі нові ваги генеруються випадковим чином. Вхідні дані для нейронної мережі наведено у таблиці 4.1.

5.6 Висновки

У даному пункті була представлена основна структура нейронної мережі, опис її файлів. Також було описано процедуру навчання нейронної мережі з наведеними блок-схемами алгоритмів навчання. Навчання нейронної мережі методом зворотного ходу застосовувалося двома етапами. Перший етап це етап розрахунку відповіді нейронної мережі при даних вхідних значеннях. Другим етапом було уточнення вагів синапсисів за допомогою отриманої помилки нейронної мережі.

Також в даному пункті була наведена реалізація багатопотоковості нейронної мережі. Ця реалізація застосовується двома способами. Перший з них це використання розпаралелювання навчальної вибірки. Такий вид розпаралелювання призводить до зміни алгоритму розрахунку нейронної мережі. Тому такий метод отримання вагів потребує подальшого досліждення.

Другий метод використовує потоки для розрахунку вагів. Доля застосування багатопотоковості в такій реалізації доволі мала. Тому при використанні такого підходу прискорення нейронної мережі відбувається на малому рівні.

В даному пункті також був описаний інтерфейс програмного забезпечення та вхідні дані для налаштування нейронної мережі.

6 РЕЗУЛЬТАТИ ОБЧИСЛЮВАЛЬНИХ ЕКСПЕРЕМЕНТІВ

В даній магістерській дисертації за мету було обрано створення алгоритму розпаралелювання нейронної мережі. Для перевірки коректності навчання була створена тестова нейрона мережа, основною задачею якої було визначення кількості частинок які випромінює тіло маючи на вході дані детектора.

Дані детектора відрізнялися для різної кількості частинок таким чином, що дійсну кількість частинок складно спрогнозувати. При цьому успішним результатом навчання вважається якщо середня похибка нейронної мережі буде менша ніж у детектора.

Для створення багатопотоковості було в програмний продукт впроваджено два алгоритми розпарелелювання навчання. Для навчання були обранні однакові умови по кількості потоків та кількості ітерацій. Було обрано 8 потоків для 390 ітерацій (15 епох).

Для розрахунку використовувався процесор AMD A10-5745M APU with Radeon(tm) HD Graphics, частотою 2.1 GHz. Кількість ядер даного процесору 4.

Похибка детектору дорівнює 14,98%.

6.1 Результат нейронної мережі без використання багатопотоковості

Для розрахунку пришвидшення багатопотоковості потрібно отримати результат нейронної мережі без використання методів паралелізації. Проходження 15 епох нейронною мережею таким способом займає 47729 мілісекунд.

Оскільки результат нейронної мережі повинний вказувати на правильну кількість частинок які вилетіли з тіла то очікуваний результат при лінійному збільшенню кількості частинок із детектору буде також лінійним.

Результат відповідей нейронної мережі зображений на рисунку 6.1.

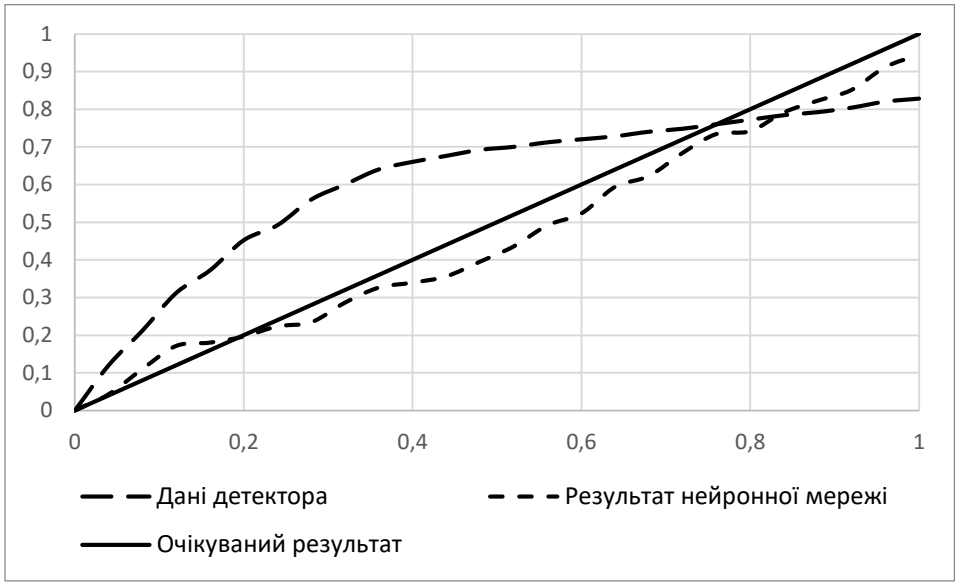

Рисунок 6.1 – Результат нейронної мережі

Похибка відносно очікуваного результату зображена на рисунку 6.2

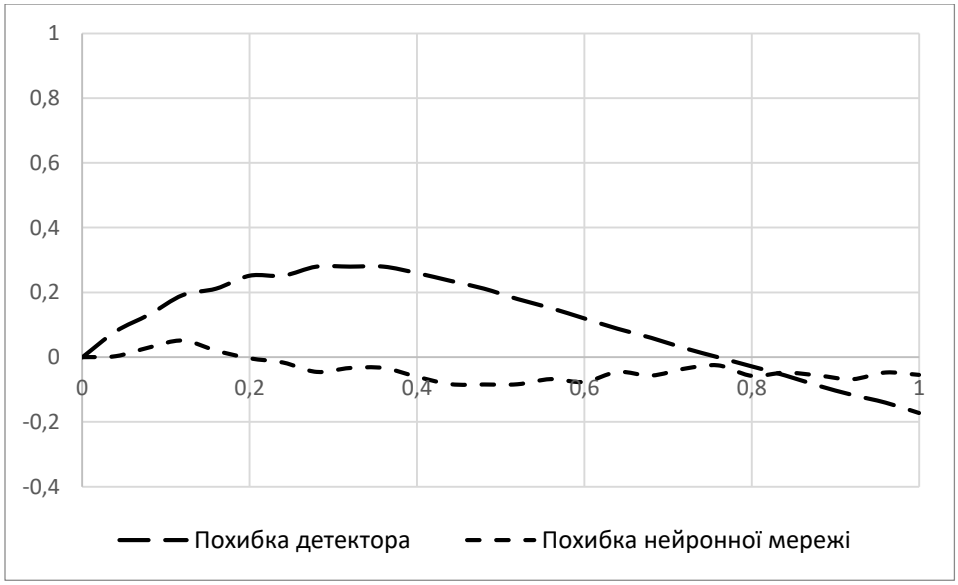

Рисунок 6.2 – Похибка відносно очікуваного результату

Таким чином похибка нейронної мережі складає 4,25%

6.2 Результати методу паралелізації на рівні вагів

Особливість даного методу полягає у відсутності можливості регулювання кількості потоків. Їхня кількість створюється в залежності від кількості синапсисів. Для забезпечення консерватизму порівняння швидкості багатопотоковості потрібно задати однакову кількість потоків. Для задання 8 потоків потрібно створити нейронну мережу із 8 синапсисами. За умови, що вхідний та вихідний шар містить лише по одному нейрону, кількість потоків можна задавати лише змінюючи кількість прихованих нейронів. Для забезпечення умови по кількості потоків, достатньо 8 прихованих нейронів. При такому налаштуванні нейронна мережа виконує завдання за 41810 мс.

При змінні кількості прихованих нейронів змінюється кількість синапсисів, що впливає на швидкість навчання нейронної мережі незалежно від наявності багатопотоковості.

Порівняння швидкості навчання нейронних мереж із даним методом багатопотоковості та без зображено на рисунку 6.3.

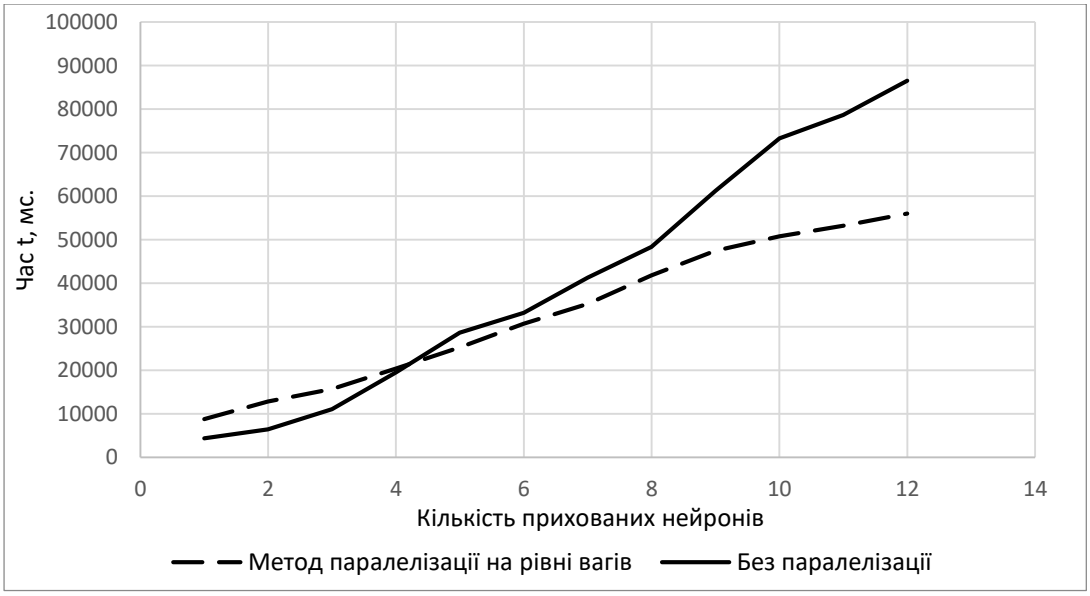

Рисунок 6.3 – Залежність швидкості навчання нейронних мереж в залежності від кількості прихованих нейронів

Із рисунку 6.3 видно, що збільшення кількості прихованих нейронів на швидкість навчання впливає менше для методу паралелізації на рівні вагів. Це пов’язано із розпаралелюваням синапсисів, тому із їх збільшенням створюється більша кількість потоків.

Результат нейронної мережі із паралелізацією на рівні вагів представлено на рисунку 6.4.

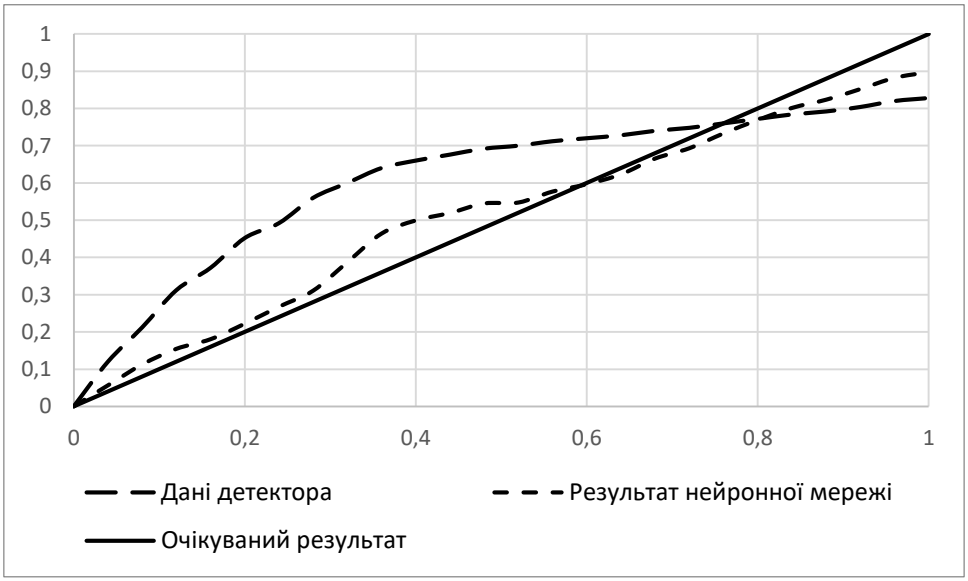

Рисунок 6.4 – Результат нейронної мережі

Похибка відносно очікуваного результату зображена на рисунку 6.5

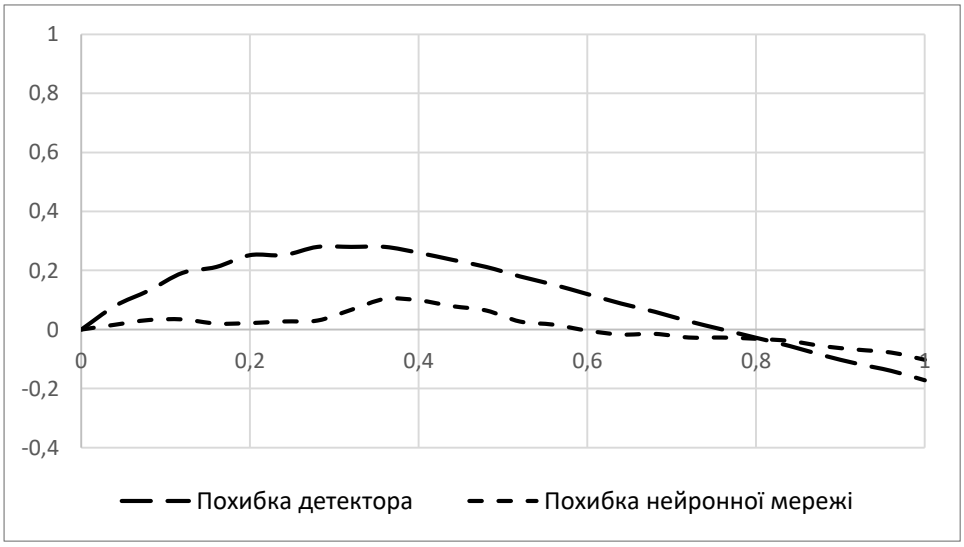

Рисунок 6.5 – Похибка відносно очікуваного результату

Для даної нейронної мережі похибка складає 4,58%. Максимальний коефіцієнт прискорення багатопотоковості дорівнює 1,61. Коефіцієнт ефективності багатопотоковості дорівнює 0,4025.

6.3 Результати методу паралелізації із динамічною кількістю потоків

Особливістю даного методу являється незалежність кількісті потоків від нейронної мережі. Це число задає користувач. Для нейронної мережі із використанням методу паралелізації із усередненням вагів зміна кількості потоків не залежить від кількості нейроннів, тому їх кількість буде постійною. Таким чином ускладнання навчання нейронної мережі не відбувається. Тому час затрачений нейронною мережею без використання багатопотоковості буде однаковий.При використанні цієї нейронної мережі із 8 прихованими нейронами та 8 потоками, час затрачений на навчання дорівнював 14604 мс.

Час затрачений на проведення навчання нейронної мережі без ускладення зображений на рисунку 6.6.

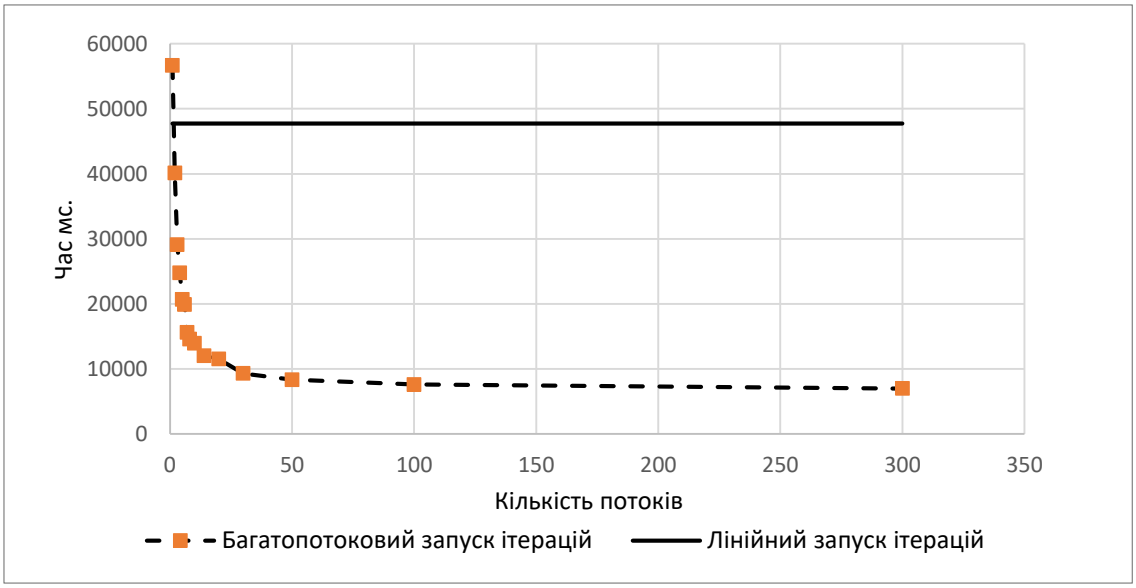

Рисунок 6.6 – Час затрачений на навчання нейронної мережі в залежності від кількості потоків

На графіку зображено дві криві. Суцільна характеризує час затрачений при лінійній оброці 390 ітерацій нейронною мережею. Оскільки для такої нейронної мережі не використовувався принцип багатопотоковості, вона пледставляє з себе горизонтальну пряму. Пунктирна крива характеризує зміну часу який потрібний для нейронної мережі, яка використовує принцип багатопотоковості, для походження 390 ітерацій залежно від зміни кількості потоків. Дані результати показали, що використання багатопотоковості тільки для одного потоку не є доцільним оскільки принцип роботи такий самий як і у лінійному запуску ітерацій а також через велику кількість операцій, таких як записування усіх синапсисів для кожного потоку а потім усередення їх, часу для розвязку 390 ітерацій потрібно більше.

При використанні 2-х та більше потоків швидше виконують завдання нейронні мережі із багатопоковістю. Після 50 потоків спад функції зменьшується через великі кількість переходів між потоками. А при 400 та вище потоків крива функції починає підніматись. Таким чином оптимальна кількість потоків лежитьв межах 50-300.

Результат нейронної мережі представлено на рисунку 6.6.

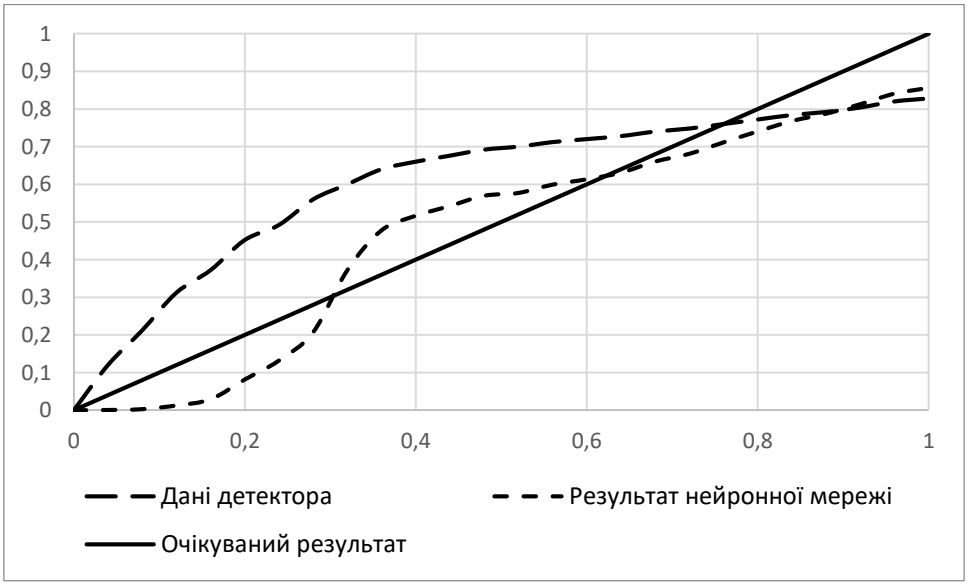

Рисунок 6.6 – Результат нейронної мережі

Похибка відносно очікуваного результату зображена на рисунку 6.7

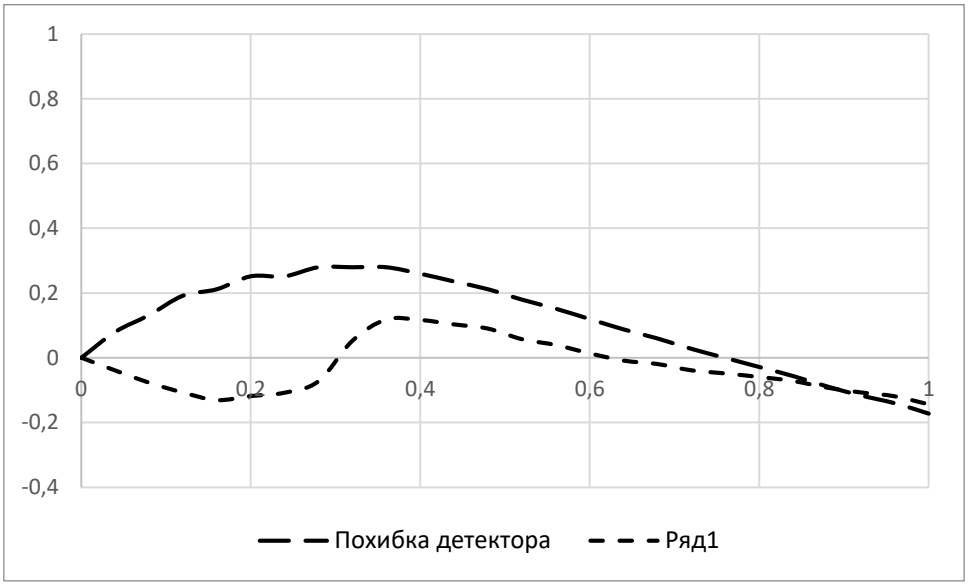

Рисунок 6.7 – Похибка відносно очікуваного результату

Для даної нейронної мережі похибка складає 7,27%. Максимальний коефіцієнт прискорення багатопотоковості дорівнює 6,82. Коефіцієнт ефективності багатопотоковості дорівнює 1,7.

6.4 Висновки

У даному розділі були наведені результати нейронних мереж які використовували два методи багатопотоковості. Окрім цього була проаналізована нейронна мережа без застосування методів паралельності.

Нейронна мережа лінійного навчання виконала 15 епох за 47729 мс. З цим самим завданням із використанням багатопотоковості методом паралелізації вагів нейронна мережа справилася за 41810 мс. Найкращий результат був отриманий нейронною мережею із фіксованою кількістю потоків, затрачений час дорівнює 14604 мс. Усі результати нейронних мереж показують їх схильність до навчання.

7 РОЗРОБКА СТАРТАП ПРОЕКТУ

Стартап – це нещодавно створена компанія, яка знаходиться лише на стадії розвитку, але має інноваційні ідеї. Саме ці нововведення допоможуть перевершити конкурентів та вийти на ринок з новим продуктом, послугою. Основна особливість стартапів – недостатність фінансів, пошук «бізес-ангелів» та відношення не тільки до ІТ-сфери, а і до будь-якої області ринку. Зараз стартап-проекти можна називати венчурними [31].

Стартап як форма малого ризикового (венчурного) підприємництва впродовж останнього десятиліття набула широкого розповсюдження у світі через зниження бар'єрів входу в ринок (із появою Інтернету як інструменту комунікацій та збуту стало простіше знаходити споживачів та інвесторів, займатись пошуком ресурсів, перетинати кордони між ринками різних країн), і вважається однією із наріжних складових інноваційної економіки, оскільки за рахунок мобільності, гнучкості та великої кількості стартап-проектів загальна маса інноваційних ідей зростає.

Проте створення та ринкове впровадження стартап-проектів відзначається підвищеною мірою ризику, ринково успішними стає лише невелика частка, що за різними оцінками складає від 10% до 20%. Ідея стартап-проекту, взята окремо, не вартує майже нічого: головним завданням керівника проекту на початковому етапі його існування є перетворення ідеї проекту у працюючу бізнес-модель, що починається із формування концепції товару (послуги) для визначеної клієнтської групи за наявних ринкових умов [32].

Розділ має на меті проведення маркетингового аналізу стартап проекту задля визначення принципової можливості його ринкового впровадження та можливих напрямів реалізації цього впровадження. Проведення маркетингового аналізу передбачає виконання нижченаведених кроків.

7.1 Опис ідеї проекту

В межах підпункту слід проаналізувати та подати у вигляді таблиць:

- зміст ідеї (що пропонується);
- можливі напрямки застосування;
- основні вигоди, що може отримати користувач товару;
- чим відрізняється від існуючих аналогів та замінників.

Перші три пункти подаються у вигляді таблиці (таблиця 6.1) і дають цілісне уявлення про зміст ідеї та можливі базові потенційні ринки.

Таблиця 7.1 - Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Розроблено метод багатопотоковості для нейронної мережі.	Створення методів моделювання	Отримання нових підходів для вирішення завдання
	Пошук прихованих залежностей	Розкриття суті завдання
	Класифікація даних	Можливість сортування даних для швидкого доступу

Аналіз потенційних техніко-економічних переваг ідеї (чим відрізняється від існуючих аналогів та замінників) порівняно із пропозиціями конкурентів передбачає:

1) визначення переліку техніко-економічних властивостей та характеристик ідеї;

2) визначення попереднього кола конкурентів (проектів-конкурентів) або товарів-замінників чи товарів-аналогів, що вже існують на ринку, та проводиться збір інформації щодо значень техніко-економічних показників для ідеї власного проекту та проектів-конкурентів відповідно до визначеного вище переліку;

3) проводиться порівняльний аналіз показників: для власної ідеї визначаються показники, що мають а) гірші значення (W, слабкі); б) аналогічні (N, нейтральні) значення; в) кращі значення (S, сильні) (таблиця 4.2).

Визначений перелік слабких, сильних та нейтральних характеристик та властивостей ідеї потенційного товару є підґрунтям для формування його конкурентоспроможності.

Таблиця 7.2 – Визначення сильних, слабких та нейтральних характеристик проекту

№ п/п	Техніко-економічні характеристики ідеї	(Потенційні) товари/концепції конкурентів		W (слабка сторона)	N (нейтр. сторона)	S (сильна сторона)
		Мій метод	Інші методи			
1	2	3	4	5	6	7
1.	Швидкість навчання	За рахунок малої кількості зв’язків між потоками	За рахунок великої кількості зв’язків між потоками			+
2.	Адаптованість під потужності характеристики системи	Можливе настроювання кількості потоків	Кількість потоків залежить від нейронної мережі			+
3.	Збереження конфігурації нейронної мережі	Можливе настроювання кількості потоків	Неможливе настроювання кількості потоків			+

Продовження таблиці 7.2

1	2	3	4	5	6	7
4.	Коректність навчання	Змінений алгоритм навчання	Незмінений алгоритм навчання	+		

7.2 Технологічний аудит ідеї проекту

Було проведено аудит технології, за допомогою якої можна реалізувати ідею проекту (технології створення товару). Визначення технологічної здійсненності ідеї проекту передбачає аналіз таких складових (таблиця 4.3):

1) за якою технологією буде виготовлено товар згідно ідеї проекту?
2) чи існують такі технології, чи їх потрібно розробити/доробити?
3) чи доступні такі технології авторам проекту?

За результатами аналізу таблиці зроблено висновок щодо можливості технологічної реалізації проекту. Технологічним шляхом реалізації проекту було обрано такі технології, як Windows Forms на платформі .NET Framework через їх доступність та безкоштовність.

Таблиця 7.3 – Технологічна здійсненність проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Створення програмного забезпечення для паралельного навчання нейронної мережі	Алгоритми навчання нейронної мережі	Наявна	Доступна
		Алгоритми розпаралелювання нейронної мережі	Наявна	Доступна
		Windows Forms	Наявна	Доступна

7.3 Аналіз ринкових можливостей запуску проекту

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

Спочатку було проведено аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку (таблиця 7.4).

Таблиця 7.4 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж, грн/ум.од	268300
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність вхідних обмежень (вказати характер обмежень)	-
5	Специфічні вимоги до стандартизації та сертифікації	-
6	Середня норма рентабельності в галузі (або по ринку), %	42

Середню норму рентабельності в галузі було порівняно із банківським відсотком на вкладення. Останній є меншим, тому є сенс вкладати гроші саме у цей проект.

За результатами аналізу таблиці 7.4 було зроблено висновок, що ринок є привабливим для входження.

Надалі були визначені потенційні групи клієнтів, їх характеристики та зформовано орієнтовний перелік вимог до товару для кожної групи (таблиця 4.5).

Таблиця 7.5 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1.	Програмне забезпечення для створення нейронних мереж які можуть навчатися методом багатопотоковості	Банківські системи, економічні та екологічні структури, інженерні структури	Мета використання програмного забезпе-чення: для пришвидшення процесу навчання нейронної мереіж	Зручний інтерфейс, точність результатів, швидкість, надійність у вико-ристанні

Після визначення потенційних груп клієнтів було проведено аналіз ринкового середовища: складено таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають (таблиці 7.6, 7.7).

Таблиця 7.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	2	3	4
1	Конкуренція	Вихід на ринок продуктів з кращими характеристиками	Передбачити додаткові переваги власного програмного продукту (ПП) для того, щоб повідомити про них саме після виходу на ринок конкурентів. Вдосконалення технічних моментів власного продукту.

Продовження таблиці 7.6

1	2	3	4
2	Невідповідність умовам технологічного розвитку	Динамічний розвиток технологій, що призведе невідповідності ПП використовуваним у сучасних приміщеннях технологіям	Забезпечення гнучкості математичних моделей, адаптація до сучасних умов швидкими темпами
3	Зміна потреб користувачів	Користувачам необхідне програмне забезпечення з іншим функціоналом	Передбачити можливість додавання нового функціоналу до створюваного ПП

Таблиця 7.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	2	3	4
1	Конкуренція	Відсутність аналогічного продукту для вітчизняного користувача.	Адаптація програмного продукту до вітчизняних особливостей.
2	Поява нових методів навчання	З’являться нові методи навчання, що дозво-лятимуть враховувати більше факторів, що впливають на швидкість навчання нейронної мережі	Покращити ПП додаванням нового функціоналу, розширення можли-востей

Продовження таблиці 7.7

1	2	3	4
3	Поява нових методів розпаралелювання	З’являться нові методи, що будуть швидше та точніше розпаралелювати навчання нейронної мережі	Покращити ПП додаванням нового функціоналу, розширення можли-востей

Надалі було проведено аналіз пропозиції: визначено загальні риси конкуренції на ринку (таблиця 7.8).

Таблиця 7.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1	2	3
1. Конкуренція: чиста	На ринку присутні декілька компаній-конкурентів, але їх товар дещо відрізняється між собою.	Підтримка якості продукту та постійні нововведення, вдосконалення.
2. Рівень конкурентної боротьби: національний	Компанії-конкуренти національного рівня	Створити ПП з урахуванням потреб користувачів у нашій країні
3. Галузева ознака: внутрішньогалузева	Продукт може використовуватись у теплотехнічній галузі	Постійне вдосконалення продукту з метою його застосовності у інших сферах

Продовження таблиці 7.8

4. Конкуренція за видами товарів: товарно-видова	Конкуренція між видами ПП, їх особливостями.	Створити ПП, враховуючи недоліки конкурентів
5. Характер конкурентних переваг: нецiновий	Вдосконалення технології створення ПП, для зниження ціни і розширення функціональності	Удосконалення моделі. Використання більш дешевих тех.нологій для розробки, ніж використовують конкуренти, але тільки за їх відповідності вимогам якості.
6. Інтенсивність: не марочна	Бренд присутній, але його роль незначна	Реклама, участь у конференціях, семінарах.

Було проведено аналіз конкуренції у галузі за моделлю М. Портера (таблиця 7.9).

За результатами аналізу таблиці 7.9 було зроблено висновок про можливість роботи на ринку з огляду на конкурентну ситуацію. Також було зроблено висновок щодо характеристик, які повинен мати проект, щоб бути конкурентоспроможним на ринку.

Цей висновок було враховано при формулюванні переліку факторів конкурентоспроможності далі. На основі аналізу конкуренції, наведеного в таблиці 7.9, а також з урахуванням характеристик ідеї проекту (таблиця 7.2), вимог споживачів до товару (таблиця 7.5) та факторів маркетингового середовища (таблиці 7.6, 7.7) визначається та обґрунтовується перелік факторів конкурентоспроможності. Аналіз оформлено у таблиці 7.10.

Таблиця 7.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Навести перелік прямих конкурентів	Визначити бар'єри входження в ринок	Визначити фактори сили постачальників	Визначити фактори сили споживачів	Фактори загроз з боку замінників
	Amazon, Google	Наявність вже існуючих рішень	-	Контроль якості продукту	Наявність більш широкого функціоналлу, автори-тет (перевіре-на якість)
Висновки:	Досить ін.-тенсивна конкурентна боротьба з гравцями, що вже закріпилися на ринку	Є можливості виходу на ринок, але є і конкуренти. Приблизний термін – 18 місяців.	-	Зручний інтерфейс, надійний, швидкий, точний та достовірний ПП для навчання нейронної мережі	Необхідно ви-пускати ПП на рівні з конкурентами та розширювати функціонал.

Таблиця 7.10 – Обгрунтування факторів конкурентоспроможності

№ п/п	Фактор конкуренто-спроможності	Обґрунтування (перелік чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Ціна	Доступніша ціна збільшує кількість потенційних клієнтів
2	Орієнтованість на кінцевого користува-ча	Продукт орієнтований на взаємодію з клієнтом
3	Адаптованість під конструкторські особливості конкретної нейронної мережі	Широке коло напрямків, для яких може бути використаний ПП

За визначеними факторами конкурентоспроможності (таблиця 7.10) проведено аналіз сильних та слабких сторін стартап-проекту (таблиця 7.11).

Таблиця 7.11 – Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з розроблюваним проектом						
			–3	–2	–1	0	+1	+2	+3
1	Ціна	8			+				
2	Орієнтованість на кінцевого ко-ристувача	12					+		
3	Адаптованість під конструкторські особливості конкретної нейронної мережі	20			+				

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) (таблиця 7.12) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін (таблиця 7.11). Перелік ринкових загроз та ринкових можливостей було складено на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Ринкові загрози та ринкові можливості є наслідками (прогнозованими результатами) впливу факторів, і, на відміну від них, ще не є реалізованими на ринку та мають певну ймовірність здійснення. Наприклад: зниження доходів потенційних споживачів – фактор загрози, на основі якого можна зробити прогноз щодо посилення значущості цінового фактору при виборі товару та відповідно, – цінової конкуренції (а це вже – ринкова загроза).

Таблиця 7.12 – SWOT-аналіз стартап-проекту

Сильні сторони: Ціна Орієнтованість на кінцевого користувача Адаптованість під конструкторські особливості конкретного приміщення	Слабкі сторони: Змінений алгоритм навчання
Можливості: Конкуренція Поява нових методів розрахунку Поява нових методів розпаралелювання	Загрози: Невідповідність умовам технологічного розвитку Зміна потреб користувачів

На основі SWOT-аналізу було розроблено альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок (див. таблицю 7.9, аналіз потенційних

конкурентів). Визначені альтернативи були проаналізовані з точки зору строків та ймовірності отримання ресурсів (таблиця 7.13).

Таблиця 4.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Терміни реалізації
1	Безкоштовне розповсюдження створеного ПП	15%	10 місяців
2	Створення ПП з подальшим розповсюдженням за певну оплату	76%	21 місяців
3	Створення вебсайту, на якому можна буде користуватися ПП	63%	14 місяців

Для подальшої реалізації було обрано альтернативу №2.

7.4 Аналіз ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: було проведено опис цільових груп потенційних споживачів (таблиця 7.14).

За результатами аналізу потенційних груп споживачів було обрано цільові групи, для яких буде запропоновано товар та визначено стратегію охопленя ринку – стратегію диференційованого маркетингу (компанія працює з декількома сегментами).

Таблиця 7.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієн-тів	Готовність споживачів сприйняти продукт	Орієнтовний попит в ме-жах цільової групи (сег-менту)	Інтенсив-ність кон-куренції в сегменті	Простота входу у сегмент
1	Компанії, діяльність яких пов’язана з пошуком необхідних залежностей в завданні	Висока	Середній рівень	Сильна	Середній рівень
2	Архітектори методу розпаралелювання	Висока	Високий	Помірна	Сильна
3	Розробники програмного забезпечення для швидкого навчання нейронної мережі	Помірна	Помірна	Висока	Складно

Для роботи в обраних сегментах ринку сформовано базову стратегію розвитку (таблиця 7.15).

Таблиця 7.15 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Створення ПП з подальшим розповсюджу	Визначити потреби кожної з груп, розробити відповідно до них стратегії	Цінова політика, універсальність продукту, орієнтова-ність на кінцевого ко-ристувача	Стратегія диферен-ціації

Наступним кроком обрано стратегію конкурентної поведінки (таблиця 7.16).

Таблиця 7.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкуренттів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Ні	Шукати нових	Ні	Стратегія заняття конкурентної ніші

На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту (див. таблицю 7.5), а також в залежності від обраної базової стратегії розвитку (таблиця 7.15) та стратегії конкурентної поведінки (таблиця 7.16) розроблено стратегію позиціонування (таблиця 7.17), що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект.

Таблиця 7.17 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту
1	Легкість розуміння, зручний інтерфейс, швидкий ПП	Стратегія диференціації	Позиція на основі порівняння фірми з товарами конкурентів;	Економія часу; Зручність застосування; Практичність та точність результату

В результаті отримано узгоджену систему рішень щодо ринкової поведінки стартап-компанії.

7.4 Розроблення маркетингової програми стартап-проекту

Сформовано маркетингову концепцію товару, який отримає споживач. Для цього у таблиці 7.18 підсумовано результати попереднього аналізу конкурентоспроможності товару. Концепція товару - письмовий опис фізичних та інших характеристик товару, які сприймаються споживачем, і набору вигод, які він обіцяє певній групі споживачів.

Таблиця 7.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Зручність застосування	Не вимагається специфічних знань для використання ПП	Використання ПП вимагає від користувача лише коректного задання інформації навчальної вибірки у файлі даних
2	Практичність результату	Підвищення швидкості навчання нейронної мережі	Підвищення швидкості навчання нейронної мережі за рахунок багатопотоковості
3	Незалежність від конструкції нейронної мережі	Гнучкість використання для різних нейронних мереж	ПП не має прив’язки до конкретного типу нейронної мережі та може бути застосованим для широкого нейронних мереж

Розроблено трирівневу маркетингову модель товару: уточняється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання (таблиця 7.19).

1-й рівень При формуванні задуму товару вирішується питання щодо того, засобом вирішення якої потреби і/або проблеми буде даний товар, яка його основна вигода. Дане питання безпосередньо пов'язане з формуванням технічного завдання в процесі розробки конструкторської документації на виріб.

2-й рівень Цей рівень являє собою рішення того, як буде реалізовано товар на ринку; включає в себе якість, властивості, дизайн, упаковку, ціну.

3-й рівень Товар з підкріпленням (супроводом) - додаткові послуги та переваги для споживача, що створюються на основі товару за задумом і товару в реальному виконанні (гарантії якості, доставка, умови оплати тощо).

Таблиця 7.19 – Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="3">Зменшення витрат на навчання нейронної мережі за рахунок пришвидшення навчання</td></tr>
<tr><td rowspan="6">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>1. Функція розрахунку відповіді нейронної мережі
2. Функція навчання нейронної мережі</td><td></td><td></td></tr>
<tr><td colspan="3">Якість: достовірність навчання нейронної мережі</td></tr>
<tr><td colspan="3">Пакування: відсутнє</td></tr>
<tr><td colspan="3">Марка: «FastNeuralNetwork»</td></tr>
<tr><td rowspan="2">III. Товар із підкріпленням</td><td colspan="3">До продажу: відсутнє</td></tr>
<tr><td colspan="3">Після продажу: оновлення методу багатопотоковості</td></tr>
<tr><td colspan="4">Відкритий доступ до вихідного коду та математичної моделі надано не буде. Ідею буде захищено патентом</td></tr>
</table>

Після формування маркетингової моделі товару слід відмітити, що проект буде захищено від копіювання шляхом патентування. Наступним кроком є визначення цінових меж (таблиця 7.20), якими необхідно керуватись при встановленні ціни на потенційний товар (остаточне визначення ціни відбувається під час фінансово-економічного аналізу проекту), яке передбачає аналіз ціни на товари-аналоги або товари субститути, а також аналіз рівня доходів цільової групи споживачів. Аналіз проведено експертним методом. В результаті проведення аналізу встановлено верхню та нижню межі встановлення ціни на товар, що надається користувачам.

Таблиця 7.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	3000$	2200$	Перша група має порівняно вищий рівень доходів за інші	1000$ (конфігурація ПП не відрізняється для різних груп споживачів, тому ціна не має діапазону коливання)

Наступним кроком є визначення оптимальної системи збуту (таблиця 7.21).

Таблиця 7.21 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Цільові клієнти – компанії та люди, які шукають способи швидкого навчання нейронних мереж.	Встановлення контактів із споживачами і підтримання їх. Формування попиту, стимулювання збуту.	Один (від виробника одразу споживачу).	Прямий канал збуту до споживача, мінімізувати збутові витрати.

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (таблиця 7.22).

Таблиця 7.22 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Цільові клієнти – компанії та люди, які шукають способи швидкого навчання нейронних мереж.	Конференції, інтернет-конференції, семінари, огляд професійної літератури, інтернет, періодичні видання у різноманітних (профільних) галузях.	Позиція на основі порівняння фірми з товарами конкурентів; Особливості потреб споживачів	створення репутації фірми-виробника; збільшення чистого прибутку та рентабельності фірми; збільшення потоків покупців та обсягів продажу;	Концепція полягатиме у апеляції до неефективності та дороговизні використання нейронних мереж звичайного методу навчання

Результатом аналізу стала ринкова (маркетингова) програма, що включає в себе концепції товару, збуту, просування та попередній аналіз можливостей ціноутворення, спирається на цінності та потреби потенційних клієнтів, конкурентні переваги ідеї, стан та динаміку ринкового середовища, в межах якого впроваджено проект, та відповідну обрану альтернативу ринкової поведінки.

7.5 Висновки

У даному розділі магістерської дисертації виконані наступні завдання:

1. Проведено аналіз програмного продукту з позиції перспективи розвитку стартап-проекту.
2. Проведено аналіз ринку існуючих рішень та потреб потенційних користувачів.
3. Вивчено потенційні можливості та ризики розвитку бізнесу за стартап-проекту.
4. Запропоновано межі ціно формування на розроблений продукт.
5. Визначено стратегії просування стартап-проекту.
6. Розроблено маркетингову програму.

ВИСНОВКИ

У даній магістерській дисертації був розроблений алгоритм паралельного навчання нейронної мережі. Були продемонстровані його переваги та недоліки.

У першому розділі дипломної роботи були представлені основні відомості про нейрону мережу. Основною її перевагою є пошук залежностей та їх класифікація.

Наступний розділ описує алгоритми навчання нейронної мережі їх переваги та недоліки. Вивчивши кожен із алгоритмів був підібраний найефективніший для розпаралелювання, оскільки не всі алгоритми піддаються багатопотоковості.

У третьому розділі були наведені основні методи розпаралелювання обраного методу навчання. При визначенні методу навчання відбувається побудова алгоритму паралелізації. Для цього необхідно розділити алгоритм навчання на етапи між якими існують точки обміну інформацією, так звані точки синхронізації, кількість яких повинна бути якомога меншою.

У четвертому розділі було представлено основні методи реалізації програмного продукту. Був обраний для розпаралелювання класс thread та для перевірки коректності навчання була створена тестова нейронна мережа. Задача нейронної мережі полягає у визначенні кількості частинок які випромінює тіло маючи на вході дані детектора. Успішним результатом навчання вважається якщо середня похибка нейронної мережі буде менша ніж у детектора.

У п’ятому розділі була описана основна частина програмної реалізації. Було описано два алгоритми розпаралелювання.

У шостому розділі були наведені результати нейронних мереж які використовували різні методи розпаралелювання, та проведення порівняння їх із нейронною мережею без використання методів паралелізації. Похибка нейронних мереж бла нижчою за похибку детектора. Тому результати нейронних мереж є прийнятними. Використання багатопотоковості з фіксованою кількості потоків змінює алгоритм навчання нейронної мережі, тому даний алгоритм потрібно досліджувати в подальшому.

ПЕРЕЛІК ПОСИЛАНЬ

1. Борисов Ю.И. Нейросетевые методы обработки информации и средства их программно-аппаратной поддержки / Борисов Ю.И, Кашкаров В.М., Сорокин С.А. // Открытые системы. – 1997.– № 4. – С. 38 – 40.

2. Горбань А.Н. Нейронные сети на персональном компьютере / А.Н. Горбань, Д.А. Россиев. – Новосибирск: Наука, 2006. – 230 с.

3. Воеводин В . Параллельные вычисления / Воеводин В .В., Воеводин Вл . В . – СПб .: БХВ – Петер - бург , 2002. – 608 с

4. Hong S.G., Kim S.W. and Lee J.J., 2015. The Minimum Cost Path Finding Algorithm Using a Hopfield Type Neural Network, Proceedings IEEE International Conference on Fuzzy Systems 4, 719–726.

5. Simon Haykin . Neural Network a comprehensive foundation(2nd edition) / Simon Haykin - Prentice Hall, 842 pages, 2013

6. Лисе А.А., Степанов М.В. Нейронные сети и нейрокомпьютеры. / А.А. Лисе, М.В. Степанов // Учеб. пособие. ГЭТУ. - СПб., 2009. 64 с.

7. Царегородцев В.Г. Перспективы распараллеливания программ снейросетевого анализа и обработки данных / В.Г. Царегородцев // Материалы cIII Всерос. конф. «Математика, информатика, управление – 2008». – Иркутск, с2014. – С. 110-117.

8. Рутковская Д. Нейронные сети, генетические алгоритмы и нечеткие системы / Д. Рутковская, М. Пилиньский, Л. Рутковский. - М. Горячая линия-Телеком 2004. – 112 с.

9. Тархов Д.А. Нейронные сети. Модели и алгоритмы. – М.: Радиотехника, 2010. – 82 с.

10. Горбань А.Н., Россиев Д.А. Нейронные сети на персональном компьютере. / А.Н. Горбань, Д.А. Россиев – Н.: Наука, 2006. – 276с.

11. Asanovic, Krste et al. (Jan 18, 2016). The Landscape of Parallel Computing cResearch: A View from Berkeley University of California, Berkeley. Technical cReport No. UCB/EECS-2016-183

12. Боголюбов Д. П., Чанкин А. А., Стемиковская К. В. Реализация алгоритма собучения самоорганизующихся карт Кохонена на графических процессорах с// Промышленные АСУ и контроллеры. 2012. № 10. С. 30-35

13. Barbara Chapman, Gabriele Jost, Ruud van der Pas. Using OpenMP: portable shared memory parallel programming (Scientific and Engineering Computation). Cambridge, Massachusetts: The MIT Press., 2008. - 353 pp

14. Kohonen T. “The self-organizing map”, Proceedings of the Institute of Electrical and Electronics, 1990, vol. 78, p. 1464 – 1480

15. Nordstrom T. Designing parallel computers for self-organizing maps. Forth Swedish Workshop on Computer System Architecture, Linkoping, 1992.

16. Боголюбов Д. П., Чанкин А. А., Стемиковская К. В. Реализация алгоритма обучения самоорганизующихся карт Кохонена на графических процессорах // Промышленные АСУ и контроллеры. 2012. № 10. С. 30-35

17. Сандерс Дж., Кэндрот Э. Технология CUDA в примерах: введение в программирование графических процессоров: Пер. с англ. – М.: ДМК Пресс, 2011.

18. Старченко А. В. и др. «Практикум по методам параллельных вычислений» Учебник/Под ред. А.В. Старченко. - М.: Издательство Московского университета, 2010

19. Хайкин С. Нейронные сети: полный курс, 2-е издание. : Пер. с англ. – М.: Издательский дом «Вильямс», 2008

20. В.В. Круглов, М.И. Дли, Р.Ю. Голунов "Нечеткая логика и искусственные нейронные сети": учебное пособие для студентов вузов, обучающихся по специальности "Прикладная информатика". - М.: Физматлит, 2001

21. Форсайт Д.А., Понс Ж. Компьютерное зрение. Современный подход. - М.: Вильямс, 2004. – 928 с.

22. Уоссермен Ф. Нейрокомпьютерная техника: Теория и практика. - М.: Мир, 1992. – 184 с

23. Флах П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных. – М.: ДМК Пресс, 2015. – 400 с.

24. Хайкин С. Нейронные сети: полный курс, 2-е издание. – М.: Вильямс, 2008. – 1103 с.

25. McCulloch Warren S, Pitts Walter. A logical calculus of the ideas immanent in nervous activity // The bulletin of mathematical biophysics. — 1943. — Vol. 5, no. 4. — Pp. 115–133.

26. В.А. Терехов, Д.В. Ефимов, И.Ю. Тюкин. Нейросетевые системы управления: Москва: Высшая школа, 2002. — 183 с.

27. А.О. Кузубов. Моделирование нейросетевой системы управления динамическим обьектом // Междунар. конф. «НАУКА НАСТОЯЩЕГО И БУДУЩЕГО». — Санкт-Петербург: ЛЭТИ СПб., 2017. — С. 100–101.

28. Hochreiter Sepp, Schmidhuber J¨urgen. Long short-term memory // Neural computation. — 1997. — Vol. 9, no. 8. — Pp. 1735–1780.

29. А.Ю. Дорогов. Теория и проектирвоание быстрых перестраиваемых преобразований и слабосаязанных нейронных сетей. — Санкт-Петербург: Политехника, 2014. — 328 с.

30. Current and future development in neural computation in steel processing / Martin Schlang, B Lang, T Poppe et al. // Control engineering practice. — 2001. — Vol. 9, no. 9. — Pp. 975–986

31. Черкасов Д. О., Сайбель Н. Ю. Стартап: характеристика понятия и этапы развития // Современное состояние и перспективы развития научной мысли: сборник статей Международной научно-практической конференции (25 мая 2015 г., г. Уфа) в 2 ч. – Уфа: АЭТЕРНА, 2015. – Ч. 1. – С.141-143.

32. Розроблення стартап-проекту: Методичні рекомендації до виконання розділу магістерських дисертацій для студентів інженерних спеціальностей / За заг. ред. О. А. Гавриша. – Київ: НТУУ «КПІ», 2016. – 28 с.

Додаток А

Застосування багатопотоковості для навчання нейронної мережі

Апробації

УКР.НТУУ “КПІ”.ТВ7132_18М

Аркушів 9

2018

Додаток Б

Застосування багатопотоковості для навчання нейронної мережі

Акт впровадження

УКР.НТУУ “КПІ”.ТВ7132_18М

Аркушів 1

2017